OPTIMAL SILO ATTACK PLAN FOR THE RED INTEGRATED STRATEGIC OFFENSIVE PLAN (RISOP)

THESIS

Richard Charles Pace
Captain, USA

AFIT/GOR/ENS/93M-15

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/93M-15

OPTIMAL SILO ATTACK PLAN FOR THE RED INTEGRATED STRATEGIC OFFENSIVE PLAN (RISOP)

THESIS

Richard Charles Pace
Captain, USA

AFIT/GOR/ENS/93M-15

AFIT/GOR/ENS/93M-15

OPTIMAL SILO ATTACK PLAN FOR THE RED INTEGRATED STRATEGIC OFFENSIVE PLAN (RISOP)

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Richard Charles Pace, B.A.
Captain, USA

March, 1993

THESIS APPROVAL

STUDENT: CPT Richard C. Pace CLASS: GOR-93M

THESIS TITLE: OPTIMAL SILO ATTACK PLAN FOR THE RED INTEGRATED STRATEGIC OFFENSIVE PLAN (RISOP)

DEFENSE DATE: March 4, 1993

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Advisor	Lt Col James T. Moore/ENS	James T. Moore
Reader	Dr. James W. Chrissis/ENS	[illegible]
Reader	Maj Bruce W. Morlan/ENC	[illegible]

Acknowledgements

I have had a great deal of help in writing this thesis. I would especially like to thank my advisor, Lt Col Moore, for always being available when I had a question and for making sure I stayed on track. Thanks is also due to my readers, Dr. Chrissis and Maj Morlan, for their timely input. I am deeply indebted to my point of contact at J-8 (DISA/DSSO), CPT Pete Davidson, who proposed this problem to begin with and who provided a tremendous amount of information and other help for the last six months. I also owe thanks to Dr. Roy Marsten of the Georgia Institute of Technology for making ZOOM available to me for this research. Finally, and most critical of all, my deepest gratitude goes to my wife, Beth, who sacrificed more than I did over the last six months in order to ensure this project was successfully completed.

Richard Charles Pace

Table of Contents

List of Figures

List of Tables

AFIT/GOR/ENS/93M-15

Abstract

The Joint Staff Directorate for Force Structure, Resources, and Assessment (J-8) sought a procedure which could be used to generate an optimal missile allocation for the silo attack portion of the Red Integrated Strategic Offensive Plan (RISOP). Their current solution procedure is a manual heuristic which is time-consuming and is not guaranteed to lead to an optimal solution. J-8 defines an optimal solution as a feasible solution which minimizes both the flight time of the missile that impacts first and the duration of the attack. J-8 defined several input rules which limit how missiles may be allocated. A mathematical model of the J-8 missile allocation problem was developed that uses a goal programming approach to solve the problem. The input rules defined the constraints for the proble. J-8 provided unclassified sample data to use as a test case. The model was used to solve the sample problem with nine different variations of the data. The model developed is a flexible tool designed to solve missile allocation problems whose objective is to minimize the first impact time or minimize the duration of the attack. The model uses binary variables, so it may be impractical to use if the number of binary variables gets large. However, in the foreseeable future, the size of the problem should decrease, thus making the model usable for at least several years.

OPTIMAL SILO ATTACK PLAN FOR THE RED INTEGRATED STRATEGIC OFFENSIVE PLAN (RISOP)

I. Introduction

1.1 Background

The problem of allocating available warheads to targets has existed since the development of long-range missiles. This problem has received much greater attention with the development of long-range, nuclear-armed missiles. As the build-up of nuclear missile arsenals in the United States and the Soviet Union progressed, so did the urgency placed on finding optimal solutions to missile allocation problems (MAPs).

Today, with the dissolution of the Soviet Union, the prevailing sentiment may be that this problem is irrelevant. However, as long as nations possess nuclear missiles, the threat of nuclear attack, although perhaps greatly diminished, still exists.

Currently, the Joint Staff Directorate for Force Structure, Resources, and Assessment (J-8) at the Pentagon generates the Red Integrated Strategic Offensive Plan (RISOP), which is used to validate the Single Integrated Operations Plan (SIOP) (Davidson, 1992a:2). The SIOP, developed by U. S. Strategic Command, is the contingency plan the United States has for the case of all-out nuclear war. The RISOP is the scenario which is believed to be the worst-case attack an enemy could conduct against the United States.

In order to generate the RISOP, J-8 uses a computer model called SINBAC (System for Integrated Nuclear Battle Analysis Calculus). SINBAC determines the optimal allocation of nuclear warheads by maximizing the expected amount of damage that an attack could inflict on the United States. Due to the target values assigned, SINBAC does not allocate any weapons for attacking U. S. missile silos. Since J-8 realizes that any adversary wishing to prevent a U. S. counterstrike will attack U. S. missile silos, they have added a routine to SINBAC that can be used to analyze a user-specified silo attack plan.

In order to generate the user-specified missile silo attack portion of the RISOP, J-8 must solve a missile allocation problem. For this MAP, J-8 designates in advance which groups of missiles from which launch fields will be used for the attack. These missiles are chosen because J-8 knows that they achieve the desired level of damage on the targets (missile silos). Thus, for this portion of their MAP, they are not concerned with maximizing the amount of damage done. Instead, J-8 has two objectives for their MAP: 1) minimize the flight time of the missile that impacts first; and 2) minimize the duration of the attack. For this MAP, "duration of the attack" is defined as the difference between the time of the first warhead impact and the time of the last warhead impact. It is assumed that the attacker would want to minimize the flight time of the first missile in order to attack the targets as soon as possible. It is assumed that the attacker would want to minimize the duration of the attack in order to limit the reaction time of the target complexes after the first complex has been attacked.

Currently, J-8 uses a manual heuristic to allocate enemy missiles against U. S. missile silos (Davidson, 1992a:2). A heuristic is a method for finding a feasible solution that is not guaranteed to be optimal. The heuristic used by J-8 basically consists of visually examining the flight times by missile type from each launch field to each target complex and assigning warheads to targets in such a way as to attempt to strike each target as early as possible. This procedure is done manually and is time-consuming.

1.2 Statement of the Problem

J-8 seeks a procedure which can be used to generate an optimal missile allocation plan; however, their current heuristic is not guaranteed to lead to an optimal solution. J-8 defines an optimal solution as a feasible solution which minimizes both the flight time of the missile that impacts first and the duration of the attack (Davidson, 1992b:2). Simply put, the problem is to develop a procedure which determines an optimal missile allocation plan. The current heuristic, which is time-consuming and does not necessarily generate an optimal solution, needs to be replaced.

1.3 Research Objectives

The major objectives of the research are two-fold. One objective is to develop a procedure which determines the optimal missile allocation plan and the other is to meet the needs of J-8. As a minimum, J-8 needs a description of how the problem may be solved using existing commercially available software as long as it can run on VAX or SUN/UNIX machines. Given this description, they want a structured algorithm describing how the problem may be solved. Finally, J-8's greatest need is for a FORTRAN routine that can be incorporated into SINBAC that determines the optimal silo attack plan (Davidson, 1992a:2).

Unfortunately, finding the "optimal" solution is not as simple as it may sound. This is due to there being two objectives for the solution of J-8's MAP that conflict somewhat: the shorter the flight time of the first missile, the longer the duration of the missile attack; and the shorter the duration, the longer the flight time of the first missile. The "optimal" solution will therefore have a flight time of the first missile that is perhaps greater than the minimum possible flight time, and a duration that is perhaps greater than the minimum that could possibly be achieved. To resolve this conflict, the decision maker must specify how short the duration should be in relation to the minimum flight time. To do this, the decision maker must define a percentage of the minimum flight time that the duration is allowed to be. For instance, if the decision maker wants the duration to be no longer than 10 percent of the minimum flight time, then if the flight time of the first missile was 40 minutes, the duration should be no longer than four minutes. In addition, the decision maker must define weighting factors associated with both the flight time of the missile which impacts first and the duration of the attack which indicate the relative importance of minimizing each of them.

1.4 Scope

In order to keep this document unclassified, J-8 has modified the data. The launch fields and target complexes are fictitious. The data for number of missiles, number of targets, distances, and flight times are also notional. In addition, missile types are identified only as "good" and "fair," and targets are not identified but only classified as "good," "fair," or "poor."

The specific missile allocation problem that J-8 needs to solve has 15 allocation rules and restrictions which more precisely define the problem. A complete discussion of these rules and restrictions and their mathematical modeling is presented in Chapter III. Below, the most restrictive rules are listed:

1. A wave (set of missiles launched at the same time) coming from one launch field is better than a wave launched from two different launch fields.
2. For fair missiles, it is better to target a given complex with two waves from the same launch field than with one wave from each of two different launch fields.
3. In the initial wave, good targets must be hit before fair targets, and fair targets must be hit before poor ones.
4. The attackers will keep at least 10 percent of their missiles as backup (strategic reserve). A "backup" missile is one that is not allocated for launch in either the initial or follow-on wave. It is best to have 10 percent of the missiles in each launch field kept as backup.

5. No "strays" are allowed. This means that the allocation should be somewhat balanced. In other words, the attackers will not allocate 90 percent of a given target complex's targets to missiles from one launch field and 10 percent to missiles from another. The allocation will be closer to 50 percent allocated to each launch field.
6. During each wave, every target in a given complex must be hit at the same time. This may result in some missiles not being launched at the same time as others. In effect, all missiles allocated to a given target complex have the same time of impact.

1.5 Description of the Unclassified Sample Problem

The data for the sample problem is given in Appendix A. Some of the launch fields have only one type of missile and the others have both types. Each target complex has only one type of target. However, J-8 has stated that Tybee and Atlanta are close enough together to be treated as a single complex. If so combined, the resulting combined complex has both good and fair targets.

The launch fields and target complexes are somewhat aggregated in that each missile of a given type from a given launch field has the same flight time to each target complex. There is no data for the flight times of each individual missile to each individual target. All flight time data is from the center of mass (centroid) of the launch field to the center of mass (centroid) of the target complex.

1.6 Approach

This problem was formulated as a mixed-integer program (MIP). The MIP's objective function is to minimize the flight time of the missile that impacts first and to minimize the duration of the attack. The other input rules establish the parameters and constraints of the problem. In formulating the problem, a constraint which ensures that the duration is no longer than a given percentage of the minimum flight time is needed.

Once the problem was formulated, a procedure was developed that solves it. This procedure was implemented in a FORTRAN program which required four major parts: one that reads in all the appropriate data, a second that processes the data and formulates the mixed-integer program (see Chapter III), a third that solves the problem, and a fourth that produces the solution in a format that J-8 can use. Ideally, this approach allows for any number of launch fields, missiles, missile types, target complexes, targets, and target types. In the solution phase, the mixed integer solver called ZOOM (Zero/One Optimization Methods) was used. This solver was chosen because it is written in FORTRAN and may be included in the program as a set of subroutines.

1.7 Format

In Chapter II, much of the literature pertaining to missile allocation problems (MAPs) is reviewed, terminology and components common to missile allocation problems are described, and solution techniques that are used to solve the problem are discussed. In Chapter III, the problem formulation and the method used to solve this problem are discussed. In Chapter IV, the results and findings are summarized. In Chapter V, conclusions and recommendations are listed. In Appendix A, the sample problem unclassified data that J-8 provided are presented. In Appendix B, the FORTRAN code which reads in the input problem data and solves the problem is listed. In Appendix C, the output files that the FORTRAN program produced in solving the sample problem are listed. In Appendix D, the GAMS (General Algebraic Modeling System) input file is presented. In Appendix E, the complete solutions and allocation summaries for each of the parametric analysis cases discussed in Chapter IV are listed. In Appendix F, the User's Guide which explains how to use the FORTRAN program to solve a missile allocation problem is presented.

II. Literature Review

2.1 Introduction

The purpose of this literature review is to examine the currently available MAP models and solution methodologies to see if one of them could be used to solve the J-8 missile allocation problem. This review first discusses the terminology and components that are common to MAP models, then discusses the specifics of the problem proposed by J-8 in terms of the components of a MAP, and finally discusses solution techniques which may be used to solve this problem.

2.2 Model Terminology

The following terms are common throughout the literature.

1. Damage Expectancy (DE): DE is the expected amount of target damage that a given weapon produces against a specific target. It is based on the probabilities that the weapon will launch, arrive, detonate, and cause various levels of damage (Seiler, 1983:11).
2. Hedge: A hedge is a constraint (input rule) that specifies side goals or limitations for allocating missiles (Bunnell and Takacs, 1984:14).
3. Target Optimization: Target optimization consists of allocating the available missiles to the targets in order to best meet the main objective. The main objective is usually to maximize the expected amount of damage done to the set of targets (Bunnell and Takacs, 1984:14).

2.3 Model Elements

All missile allocation models have five submodels in common: the weapon system, the target complex, the engagement model, the damage model, and the solution algorithm (Matlin, 1970:337). Each submodel has a certain level of complexity. In general, the more complex each submodel is, the harder it is to find the optimal solution. Thus, no model is robust enough to handle each submodel at its highest level of complexity (Matlin, 1970:337).

2.3.1 The Weapon System. The weapon system consists of three distinct categories: the scope (number of weapon types and penetration aids), weapon "reach" (which targets each weapon may strike), and weapon commitment policy (the number of waves launched, quality of bomb-damage assessment, and weapon availability uncertainties) (Matlin, 1970:337–339).

2.3.1.1 Scope. The scope of the weapon system is defined by the number of weapon types available and whether or not penetration aids are available (Matlin, 1970:339). Penetration aids are those items which help a weapon system defeat the target defensive system. These may include chaff, terminal decoys, and area decoys (Bunnell and Takacs, 1984:10). The simplest models would assume a single weapon type with no penetration aids. The most complex models allow for multiple weapon types and the use of penetration aids which are usually modeled as an overall probability of breaching the defenses and detonating (Matlin, 1970:339). Grotte (1982:430) allowed for up to four types of weapons, while the arsenal exchange model (AEM) allows for up to 25 (Bozovich and others, 1973:7). Some models express weapons in equivalent units which permits easier solution of the problem (Lemus and David, 1963:789; Wambsganss, 1982:10).

2.3.1.2 Weapon Reach. Weapon reach is usually given as an incidence matrix where a "1" entry indicates the weapon can reach a given target while a "0" entry indicates it cannot. The simplest models assume all missiles can reach all targets. More complex models consider range restrictions. The most complex models consider various payloads which missiles may carry to different ranges (Matlin, 1970:339–340, 358–360).

2.3.1.3 Weapon Commitment Policy. Weapon commitment strategies may be divided into two categories: how many waves are allowed in the model and whether the commitment is deterministic or probabilistic. Deterministic commitment means all missiles are available and launch reliably, and bomb-damage assessment is perfect. Probabilistic commitment means missile boosters may fail or enemy action may eliminate missiles before launch. The AEM allows for probabilistic commitment (Bozovich and others, 1973:IV-B-11). The simplest models, such as Day's targeting complex problem (Day, 1966:992–1013), assume a single attack wave with all missiles available. The most complex models allow for more than one attack wave with each missile having a probability of not launching (Matlin, 1970:340–341). Grotte's model (Grotte, 1982:433) went so far as to consider the possibility of a catastrophic failure for an entire segment of the weapon inventory.

2.3.2 The Target Complex. The target complex also consists of three distinct characteristics: the types of targets considered, the target values, and the defenses available to the target complex (Matlin, 1970:341).

2.3.2.1 Types of Targets. Targets may be classified as either *point* or *area*, and either *dependent* or *independent*. A point target is one that is small enough that a single weapon may destroy it. A missile silo is an example of a point target. If the target is large enough that it requires more than one weapon to destroy it, it is an area target. Military bases and cities are area targets. A dependent target is a collection of point targets or an area target that has been divided into several individual aim-points such that a single weapon could destroy more than one of them. Otherwise, point and area targets are considered independent. The simplest models consider only independent targets. The most complex models, such as Day's model (Day, 1966:993–999), consider dependent area and point targets (Bunnell and Takacs, 1984:12–13).

2.3.2.2 Target Values. The usual measure of effectiveness chosen is the expected target value killed. The simplest models consider (sometimes implicitly) that all targets have the same value. More complex models rank targets in order of priority (ordinal value scale), but do not assign numerical values. In the next level of complexity, models assign numerical values to each target (cardinal value scale). At the highest level of complexity, models allow for both targets that have indirect or intrinsic value and targets that have direct or extrinsic value. Targets with indirect (intrinsic) value are those with no value assigned for killing them, but if all such targets are killed, then targets with assigned value may be eliminated either automatically or more easily (Matlin, 1970:341–342). Examples of targets with indirect value would be surface-to-air missile sites and command-and-control facilities.

2.3.2.3 Available Target Defenses. The simplest models assume there are no target defenses. More complex models treat defenses by eliminating some of the attacking missiles before they arrive. This may be done by either having an "admission price" (the defenses *will* shoot down a given number of attacking missiles) or with a probability that an attacking missile gets through (Bunnell and Takacs, 1984:14).

2.3.3 The Engagement Submodel. The engagement submodel determines the probability that a weapon breaks through the target defenses (if present). This is based on whether the offense is deterministic or probabilistic and whether the defense is deterministic or probabilistic. Deterministic offenses have missiles that hit exactly where they are aimed and penetration aids that work perfectly. Probabilistic offenses have missiles that may not hit exactly where they are aimed and penetration aids that work imperfectly. Deterministic defenses have radars that always detect incoming missiles and anti-missile missiles (AMMs) that always work. Probabilistic defenses have imperfect radars and AMMs. The simplest models assume both the offense and the defense are deterministic. The most complex models assume both are probabilistic (Matlin, 1970:343-344).

2.3.4 The Damage Submodel. The damage submodel determines how the target damage or value accumulates as a function of the number and types of attacking missiles. Damage may be either deterministic or probabilistic and either partial or total. The simplest models assume a given number of weapons will always destroy a given target. More complex models use a probability that a given target will be destroyed by a given number of missiles. A little higher complexity level allows for partial damage to accumulate before a target is considered destroyed. The most complex models, like the CODE 50 Nuclear Exchange Model (Hillerman, 1971:67-71), allow for different weapon types, each with different probabilities for different levels of damage, and a weapon commitment strategy which permits attack of targets by several types of weapons (Matlin, 1970:344-345). Bracken and McGill use both total and partial damage (Bracken and McGill, 1973:31-32).

2.3.5 The Solution Algorithm. (Matlin, 1970:345-346). Analysts have used many different algorithms or computational procedures to solve MAPs. These include analysis (differentiating the expected damage equations, setting each to zero, and solving the resulting system of equations), game theory, graphical techniques, graph theory, linear programming, dynamic programming, nonlinear programming, exhaustive searches, Monte Carlo techniques, and combinations of these. Some key properties to consider when comparing solution techniques are:

1. whether the solution is optimal or only near-optimal, and how optimality is established;
2. whether the solution is integer or continuous;
3. whether the algorithm also determines the optimal defensive allocation; and

4. the capability of the algorithm to generate optimal solutions and the computational complexity of the algorithm.

> The ideal algorithm should provide integer solutions, yield a proven optimal solution, derive the optimum defensive allocation as well as the optimum offensive allocation, be capable of handling large weapon and target complexes, run rapidly, be insensitive to small variations in weapon and target numbers and associated parameters, and provide a global rather than a local or restricted solution. (Matlin, 1970:346)

2.4 The J-8 Missile Allocation Problem

J-8 has defined certain input rules for their specific missile allocation problem. There are a given number of fixed launch fields and a given number of target complexes. Each launch field contains a known number of missiles, and each target complex has a known number of targets. In terms of the components of a MAP listed above, the problem proposed by J-8 may be summarized as the following (Davidson, 1992b:2–3):

1. Weapon System:
    (a) Scope: There are two types of missiles: good and fair. No penetration aids are available.
    (b) Reach: All missiles may range all targets with no degradation.
    (c) Commitment Policy: There may be up to two waves: 1) an initial wave which will, providing enough resources are available, attack each target; and 2) a follow-on wave which will attack the targets that were hit with a fair missile in the initial wave. The follow-on wave will be launched 30 minutes after the initial wave is launched. All missiles are available and launch reliably.
2. Target Complex:

   (a) Types of Targets: All targets are missile silos or command-and-control stations and are considered independent point targets.

   (b) Target Values: There are three categories: good, fair, and poor (ordinal value scale).

   (c) Available Defenses: None.

3. Engagement Model: All missiles hit exactly where they are aimed.

4. Damage Model: It takes one good missile or two fair missiles to destroy any target.

5. Solution Algorithm: The current algorithm employed by J-8 is a manual heuristic which does not necessarily yield the optimal solution.

6. Other Input Rules: Within this framework, J-8 has imposed 15 allocation rules and restrictions. These rules and restrictions are addressed in Chapter III.

2.5 Solution Techniques

In formulating the J-8 missile allocation problem as a mixed-integer program (see Chapter III), two solution techniques, *goal programming* and *disjunctive constraints*, were used. These techniques are discussed below.

2.5.1 Goal Programming. (Winston, 1991:175-178). For some problems, there may be more than one objective or goal that the decision maker desires to meet. If the decision maker can determine the relative importance of meeting each goal, then goal programming may be used to solve the problem.

Assume there are n goals that the decision maker seeks to achieve. Each goal may be expressed as a function of the decision variables:

$$f_i(\mathrm{x}) = g_i \quad \text{for } i = 1, 2, ..., n$$

where g_i is the decision maker's desired value for $f_i(\mathbf{x})$. In a solution of this problem, each goal will either be met exactly, be underachieved by some amount (say S_i), or be overachieved by some amount (say O_i). Then, in modeling the problem, each goal becomes a constraint of this form:

$$f_i(\mathbf{x}) + S_i - O_i = g_i.$$

If we let W_i^+ be the relative penalty of exceeding goal i, and W_i^- be the relative penalty for being below goal i, then the objective function may be written as

$$min \sum_{i=1}^{n}(W_i^+ O_i + W_i^- S_i).$$

If there are goals that the decision maker desires to achieve or exceed (it does not matter by how much), set $W_i^+ = 0$. For those the decision maker desiies to achieve but not exceed (it does not matter how much below), set $W_i^- = 0$.

2.5.2 Disjunctive Constraints. (Garfinkel and Nemhauser, 1972:11). Consider a linear program which has a set of constraints (say m of them) of which at least k constraints must hold $(1 \leq k \leq m-1)$. In general, this type of problem would have the form:

$$max \text{ (or } min) \ \{f(\mathbf{x}) : \mathbf{x} \in \mathbf{S}\}$$

(where S is an n-dimensional space defined by the constraints of the problem) where at least k of the following constraints must hold $(1 \leq k \leq m-1)$:

$$g_i(\mathbf{x}) \geq 0 \quad \text{for } i = 1, 2, ..., m.$$

To model this situation, replace the above constraints with

$$\begin{aligned} g_i(\mathbf{x}) &\geq \delta_i g_i \quad \text{for } i = 1,2,...,m \\ \sum_{i=1}^{m} \delta_i &\leq (\text{or } =)\, m - k \\ \delta_i &= 0 \text{ or } 1 \quad \text{for } i = 1,2,...,m \end{aligned}$$

where g_i is a known, negative, finite lower bound on $g_i(\mathbf{x})$. The first set of constraints (one for each of the original set of m constraints) states that $g_i(\mathbf{x})$ will be greater than or equal to 0 if δ_i is zero and greater than or equal to g_i if δ_i is one. The second constraint states that the sum of the δ_is must be less than or equal to (or equal to) the total number of constraints in this set minus the number of constraints of this set that must hold. Therefore, at least k of the δ_is will be set to zero, which means that at least k of the original constraints must hold. The last constraint simply declares each δ_i to be a binary variable.

2.6 Conclusion

Many analysts have attempted to model the missile allocation problem. Some considered only the defender's objective, some only the attacker's objective, and some both. All of the models reviewed consider the measure of effectiveness to be some variation of the expected value of damage done to the targets. Several models could handle portions of the J-8 problem; however, none can fully model the J-8 objective: to minimize the flight time of the missile that impacts first and to minimize the duration of the attack.

III. Model Formulation and Methodology

3.1 Introduction

This chapter contains the mixed-integer programming formulation of the J-8 missile allocation problem. First, the variables and parameters of the model are defined. Then, the mathematical model of the problem is formulated. Finally, the solution methodology is presented.

3.2 Variable Definitions

In the following discussion, the variables of the J-8 missile allocation problem are defined. Let

$$\begin{aligned} LF &= \text{the number of launch fields,} \\ TC &= \text{the number of target complexes,} \\ MT &= \text{the number of missile types,} \\ TT &= \text{the number of target types.} \end{aligned}$$

Further, let

j_G = the set of target complexes that contain at least one good target,

j_F = the set of target complexes that do not contain any good targets but do contain at least one fair target,

j_P = the set of target complexes that do not contain any good or fair targets.

Define

X_{ijkl} = the number of missiles at launch field i allocated to target complex j which are of type k and attack type l targets for $i = 1,2,...,LF$; $j = 1,2,...,TC$; $k = 1,2,...,MT$; and $l = 1,2,...,TT$.

Let

$$Y_{ijk} = \begin{cases} 1 & \text{if } \sum_{l=1}^{TT} X_{ijkl} > 0 \\ 0 & \text{if } \sum_{l=1}^{TT} X_{ijkl} = 0 \end{cases} \quad \text{for every } i,j,k.$$

Thus, Y_{ijk} is a variable that indicates whether or not there are missiles allocated from launch field i to target complex j of type k.

Let

$$S_{ij} = \begin{cases} 1 & \text{if } \sum_{k=1}^{MT} Y_{ijk} > 0 \\ 0 & \text{if } \sum_{k=1}^{MT} Y_{ijk} = 0 \end{cases} \quad \text{for every } i,j$$

so that S_{ij} is a variable that indicates whether or not there are missiles allocated from launch field i to target complex j.

Let

Z_j = the time that target complex j is attacked. Z_j is greater than or equal to the largest flight time of missiles allocated to target complex j.

T_{first} = the flight time of the missile that impacts first
= the earliest time of attack of the target complexes.

T_{last} = the flight time of the missile that impacts last
= the latest time of attack of the target complexes.

Dur = the duration of the attack $= T_{last} - T_{first}$.

O_1 = the amount T_{first} is over its goal (which is T_{min}, defined below).

O_2 = the amount Dur is over its goal (which is 0).

The parameters of the problem are:

T_{ijk} = the flight time from launch field i to target complex j of a type k missile.

Pct	=	the fraction of T_{first} that Dur is allowed to be.
m_{ik}	=	the number of missiles available at launch field i of type k.
n_{jl}	=	the number of targets at target complex j of type l.
T_{max}	=	the longest flight time possible ($\max\{T_{ijk}\}$).
T_{min}	=	the shortest flight time possible ($\min\{T_{ijk}\}$).
W_1	=	the weighting factor associated with T_{first}.
W_2	=	the weighting factor associated with Dur.

3.3 Model Formulation

3.3.1 Objective Function. Since there are two objectives—mii imize the flight time of the missile that impacts first and minimize the duration of the attack—a goal programming approach is used. Since J-8 desires the flight time of the missile that impacts first to be as short as possible and the duration of the attack to be as short as possible, the goal for T_{first} is T_{min} and the goal for Dur is 0. There is no penalty for underachieving (which is impossible), but there is a penalty associated with not achieving the goals. Thus, the objective function may be written as:

$$min \;\; W_1O_1 + W_2O_2$$

where W_1 and W_2 are user-specified weighting factors associated with T_{first} and Dur, respectively, which indicate the relative importance of minimizing each of these variables (i.e., if the decision maker wants to make sure that T_{first} gets as small as possible and feels that it is not nearly as important to minimize Dur, assign W_1 a value much greater than W_2).

3.3.2 Constraints. There are several constraints on the problem. Each type of constraint is discussed below.

1. Every target must be hit by a missile in the first wave. For each target complex/target type combination, this can be enforced by requiring the total number of missiles allocated from each launch field to a target complex to be equal to the number of targets of the given type at the target complex. Mathematically, this becomes
$$\sum_{i=1}^{LF}\sum_{k=1}^{MT} X_{ijkl} = n_{jl} \text{ for every } j, l.$$

2. The allocation of more missiles of a given type from a particular launch field than that launch field has available is prohibited. For each launch field/missile type combination, this may be enforced by requiring the total number of missiles allocated from a launch field to each target complex to be less than or equal to the number of missiles of the given type available at the launch field. The appropriate constraints are:
$$\sum_{j=1}^{TC}\sum_{l=1}^{TT} X_{ijkl} \leq m_{ik} \text{ for every } i, k.$$

3. In order to determine the flight time of the missile that impacts first, the smallest flight time of the missiles that are allocated must be identified. Variable Y_{ijk} is therefore needed to indicate whether or not missiles of type k have been allocated from launch field i to target complex j. Two constraints are needed for each Y_{ijk} variable to ensure that it is assigned 1 when $\sum_{l=1}^{TT} X_{ijkl} > 0$ and 0 otherwise. The first constraint should state that for each launch field/target complex/missile type combination, the associated variable Y_{ijk} must be less than or equal to the total number of missiles of type k from launch field i allocated to target complex j. This constraint forces Y_{ijk} to be 0 if there are no missiles of type k allocated from launch field i to target complex j. The second constraint should state that the associated variable Y_{ijk} must be greater than or equal to the total number of missiles of type k from launch field i to target complex j divided by the total number of targets at target complex j.

This constraint forces Y_{ijk} to be 1 if there are missiles of type k allocated from launch field i to target complex j because the number of missiles of type k allocated from launch field i to target complex j can never be greater than the total number of missiles at target complex j. These two sets of constraints may be written as:

$$\left.\begin{array}{lcl} Y_{ijk} & \leq & \sum_{l=1}^{TT} X_{ijkl} \\ Y_{ijk} & \geq & \frac{\sum_{l=1}^{TT} X_{ijkl}}{\sum_{l=1}^{TT} n_{jl}} \end{array}\right\} \text{for every } i, j, k.$$

4. The desire that each target complex be attacked by missiles from only one launch field generates a need to determine which launch fields are attacking each target complex. This can be done by setting S_{ij} to 1 if $\sum_{k=1}^{MT} Y_{ijk} > 0$ and to 0 otherwise. To do this, two sets of constraints are needed. For every launch field/target complex combination, the first constraint should state that the associated variable S_{ij} should be less than or equal to the sum of each associated variable Y_{ijk}. This forces S_{ij} to be 0 if there are no missiles allocated from launch field i to target complex j. The second constraint should state that the associated S_{ij} must be greater than or equal to the sum of each associated Y_{ijk} divided by the number of missile types. This forces S_{ij} to 1 if there are missiles allocated from launch field i to target complex j because the sum of each associated Y_{ijk} can never be greater than the number of missile types. These two constraints may be written as:

$$\left.\begin{array}{lcl} S_{ij} & \leq & \sum_{k=1}^{MT} Y_{ijk} \\ S_{ij} & \geq & \frac{1}{MT} \sum_{k=1}^{MT} Y_{ijk} \end{array}\right\} \text{for every } i, j.$$

5. Each target complex should be attacked by missiles from only one launch field. For each target complex, this may be enforced by having a constraint that states the sum of the associated S_{ij}s must be equal to 1. This means that

exactly one launch field attacks each target complex. This constraint may be expressed as:

$$\sum_{i=1}^{LF} S_{ij} = 1 \text{ for each } j.$$

If this is infeasible, then it would be desirable for as few target complexes as possible to be attacked by two launch fields. If the right hand side of the above constraint were changed to be ≤ 2, it would allow *each* target complex to be attacked by two launch fields. Thus, a constraint would be needed to limit the number of target complexes that could be attacked by two launch fields. If each target complex were attacked by one launch field, the sum of all the S_{ij} variables would be equal to the number of target complexes (TC). Therefore, adding a constraint that forced the sum of all the S_{ij}s to be less than or equal to the number of target complexes plus one would allow at most one target complex to be attacked by two launch fields. Thus, if it is not feasible to attack each target complex with only one launch field, change the right hand side of the above constraint to ≤ 2, add the following constraint to the formulation:

$$\sum_{i=1}^{LF} \sum_{j=1}^{TC} S_{ij} \leq TC + 1,$$

and re-solve the problem. This has the effect of allowing exactly one target complex to be attacked by two launch fields. If this is also infeasible, increase the right hand side of the new constraint by one. Continue this until a feasible solution is obtained.

NOTE: The constraint to allow each target complex to be attacked by missiles from only one launch field may also be expressed in terms of only the Y_{ijk} variables, which eliminates the need for the S_{ij} variables. In terms of just the

Y_{ijk} variables, this constraint becomes

$$\sum_{i \backslash i^*} \sum_{k} Y_{ij^*k} - M(1 - Y_{i^*j^*k^*}) \leq 0 \text{ for every } i^*, j^*, k^*$$

where i^* is a particular launch field, j^* is a particular target complex, k^* is a particular missile type, and M is a large number. One of these constraints is needed for each possible combination of launch field/target complex/missile type. What these constraints do is this: if $Y_{i^*j^*k^*}$ is 1, then no other launch field may shoot missiles of any type at target complex j^*. If $Y_{i^*j^*k^*}$ is 0, and if M is big enough, then that particular constraint does not limit what the other Y_{ijk}s may be. For M to be "big enough," it needs to be at least as big as the total number of possible combinations of launch fields and missile types, not including launch field i^*.

There are advantages and disadvantages to each formulation. If the user decides to use S_{ij}s, the formulation is easier to understand, but it requires more binary variables which may cause the solution time to get very large. If the user decides to use only Y_{ijk}s, fewer binary variables are needed, so solution times should be faster. However, if a particular set of data happens to be infeasible, this constraint is not as easy to alter to allow up to two launch fields to attack each target complex.

The number of constraints must also be considered. If S_{ij}s are used, the program requires an additional $2 \times LF \times TC + TC$ constraints. Without the S_{ij}s, the program requires an additional $LF \times TC \times MT$ constraints. For the sample data, considering only the four launch fields that may be used in the allocation, and considering only feasible launch field/missile type and target complex/target type combinations, there is not much of a difference in the total number of constraints for either formulation (203 with the S_{ij}s, 185 without). However, if LF, TC, or MT is different, there could be an appreciable difference in the number of constraints required for each formulation.

6. During the initial wave, every target in a given complex must be hit at the same time. Thus, for a given allocation, the time a given target complex is attacked is greater than or equal to the flight time of the missile that takes the longest to get there. All other missiles attacking that target complex have their launch times delayed so that all missiles impact simultaneously. Since Z_j is the time target complex j is attacked, Z_j must be greater than or equal to the flight time of each missile allocated to target complex j. Z_j may be determined with this set of constraints:

$$Z_j \geq T_{ijk} Y_{ijk} \quad \text{for every } i, j, k.$$

This ensures that Z_j is greater than or equal to the flight time of missiles allocated to target complex j because if there are no missiles of type k allocated from launch field i to target complex j, then the right hand side of the above constraint is zero because the associated Y_{ijk} is zero. Similarly, if there are missiles of type k allocated from launch field i to target complex j, then Z_j is greater than or equal to T_{ijk} because Y_{ijk} is set to one.

7. Since T_{first} is the flight time of the missile that impacts first, it should be set equal to the smallest Z_j. This can be done with a set of disjunctive constraints which guarantee that T_{first} is greater than or equal to at least one of the Z_js:

$$\begin{aligned} T_{first} &\geq Z_j + \delta_j(-T_{max}) \quad \text{for every } j \\ \sum_{j=1}^{TC} \delta_j &= TC - 1 \end{aligned}$$

where δ_j is a binary variable for each target complex j and T_{max} is the maximum flight time possible (the largest T_{ijk}). The first set of constraints (one for each target complex) states that T_{first} is greater than or equal to Z_j if δ_j is zero. This is true because T_{max} is greater than or equal to all other T_{ijk}s, and is therefore greater than or equal to each Z_j. The last constraint states that the sum of the δ_js must be equal to the number of target complexes minus one, which means that only one of the first set of constraints holds (T_{first} is greater than or equal to at least one of the Z_js). Since Z_j is defined as the time that target complex j is attacked, T_{first} should be equal to the smallest Z_j. Therefore, in order to set T_{first} equal to the smallest Z_j, another set of constraints is needed to ensure that T_{first} is less than or equal to each Z_j. The appropriate constraints are

$$T_{first} \leq Z_j \quad \text{for every j.}$$

8. Since T_{last} is the flight time of the missile that impacts last, it is equal to the time when the last target complex is attacked. This means it must be set greater than or equal to each Z_j. This yields the following constraints:

   $$T_{last} \geq Z_j \quad \text{for each } j.$$

   Since part of the objective function is to minimize the duration, which is the difference between T_{last} and T_{first}, T_{last} is set as small as possible. Because of this and the above constraints, T_{last} is set equal to the greatest Z_j.

9. Good targets must be attacked before fair targets, and fair targets must be attacked before poor ones. Therefore, the time of attack (Z_j) for each target complex with good targets must be less than or equal to the time of attack for each target complex that does not contain good targets but does contain fair ones, and the time of attack for target complexes with fair targets and no good targets must be less than or equal to the time of attack for each target complex that does not contain any good or fair targets. Thus, the following

set of constraints is needed:

$$\begin{aligned} Z_{j^*} &\leq Z_{j'} \text{ for every } j^* \in j_G, j' \in j_F \\ Z_{j'} &\leq Z_{j^\dagger} \text{ for every } j' \in j_F, j^\dagger \in j_P \end{aligned}$$

where j_G is the set of target complexes that contain at least one good target, j_F is the set of target complexes that do not contain any good targets but do contain fair ones, and j_P is the set of target complexes that do not contain any good or fair targets.

10. Constraints are also needed to define the variables in the objective function in terms of T_{first} and Dur. Since O_1 and O_2 are the amounts by which T_{first} and Dur exceed their respective goals, the following constraints are needed:

$$\begin{aligned} Dur &= T_{last} - T_{first} \\ T_{first} - O_1 &= T_{min} \\ Dur - O_2 &= 0 \end{aligned}$$

where T_{min} is the minimum flight time possible (smallest T_{ijk}). The first constraint sets the duration (Dur) to be equal to the difference between when the last target complex is attacked (T_{last}) and when the first target complex is attacked (T_{first}), the second constraint sets O_1 to be equal to how much T_{first} exceeds T_{min}, and the third constraint sets O_2 to be equal to how much Dur is above zero.

11. Since J-8 also wants the duration of the attack to be no bigger than a user-specified percentage of the minimum flight time, the following constraint is needed:

$$Dur \leq Pct \times T_{first}.$$

This constraint states that Dur must be less than or equal to a user-specified percentage (Pct) of T_{first}.

12. All the input rules must be addressed in the formulation. Each input rule is presented here with a discussion of how it is addressed by the model.

    (a) Input rule #1 says there are six launch fields and seven target complexes. This is problem dependent. The model is able to accommodate different numbers of launch fields (LF) and target complexes (TC).

    (b) Input rule #2 says there are two types of missiles. Again, this is problem dependent, and the model handles different numbers of missile types (MT).

    (c) Input rule #3 says there are three types of targets. The model allows for different numbers of target types (TT).

    (d) Input rule #4 says that a wave coming from one launch field is better than a wave launched from two different launch fields. This is enforced by constraint set 5.

    (e) Input rule #5 says that it takes one good missile or two fair missiles to destroy any target, and input rule #6 says that there may be up to two waves: an initial wave and a follow-on wave which must impact 30 minutes after the initial wave impacts. This is modeled by scheduling the second wave after the initial solution is found. If a target is hit by a good missile in the initial wave, it is not targeted in the follow-on wave. If a target is hit by a fair missile in the initial wave, it is targeted with a fair missile in the follow-on wave.

    (f) Input rule #7 says that all missiles are available for launch and launch reliably. This is dealt with by not decreasing the supply of available missiles.

(g) Input rule #8 says that for fair missiles, it is better to target a given target complex with two waves from the same launch field than with one wave from each of two different launch fields. This is handled by considering only 50 percent of the fair missiles as being available for the first wave. Thus, prior to solution, for m_{ik} representing fair missiles, m_{ik} is set to half the actual number of fair missiles in each launch field i (rounding down for any fraction). This results in the second wave having an allocation of fair missiles identical to that of the first wave.

(h) Input rule #9 says that each missile carries 10 warheads and so may be targeted against up to 10 targets within the same target complex. This is modeled by using the number of missiles (not warheads) at each launch field and by dividing the number of targets (n_{jl}) at each target complex by 10 and rounding up for any fraction before solving the problem.

NOTE: It is possible that some time in the future there will be different missile types that carry different numbers of warheads. If each type of missile carries a different number of warheads, this may be handled by defining a new parameter, $Numwar_k$, which would be the number of warheads that each type k missile carries. Then each X_{ijkl} would be in terms of warheads instead of missiles, m_{ik} would be the total number of type k warheads at launch field i, and n_{jl} would be the total number of type l targets at target complex j. The problem may then be solved in the same way. The resulting solution would be in terms of warheads instead of missiles.

One difficulty this could cause is that each X_{ijkl} may not be an exact multiple of the appropriate $Numwar_k$. If this would occur, the allocation would indicate that X_{ijkl} divided by $Numwar_k$ (rounded up for fractions) missiles should be launched from launch field i to type l targets at target complex j. Any excess warheads should be indicated so that the user may allocate them to other targets in the vicinity of target complex j.

(i) Input rule #10 says that good targets must be attacked before fair targets, and fair targets must be attacked before poor ones. This rule is enforced by constraint set 9.

(j) Input rule #11 says that the attackers keep at least 10 percent of their missiles as backup (strategic reserve), and it is best to have 10 percent of the missiles in each launch field kept as backup. This percentage may change in the future. Thus, this number is treated as a parameter input by the user (*Pctbu*). The requirement for at least *Pctbu* of each launch field's missiles to be reserved as backup is handled by multiplying the number of available missiles of type k at launch field i (m_{ik}), for every i and k, by $(1 - Pctbu)$ (rounding down for any fraction) before solving the problem.

(k) Input rule #12 says that two specific launch fields are assigned to the strategic reserve. Before solving the problem, the user must input whether or not any launch fields are for backup only. If so, the user must also input how many and which ones. When solving the problem, any launch field designated as backup only are ignored.

(l) Input rule #13 says that two target complexes are sufficiently close so as to be treated as a single complex. Before solving the problem, the user must specify whether or not any target complexes should be combined. If so, the usei must indicate which ones. The program then combines them by calculating the weighted average of the missile flight times:

$$T_{ij^*k} = \frac{\sum_{l=1}^{TT} n_{j_1 l} T_{ij_1 k} + \sum_{l=1}^{TT} n_{j_2 l} T_{ij_2 k}}{\sum_{l=1}^{TT} n_{j_1 l} + \sum_{l=1}^{TT} n_{j_2 l}} \quad \text{for every } i, j^*, k$$

and by summing the number of targets at each complex:

$$n_{j^*l} = n_{j_1 l} + n_{j_2 l} \quad \text{for every } l$$

where j_1 is one of the target complexes to be combined, j_2 is the other one, and j^* represents the combination of the two target complexes. The weighted average is used because the flight times are from the centroid of the launch field to the centroid of the target complex.

(m) Input rule #14 says that no "strays" are allowed. This means that the allocation should be somewhat balanced. This is enforced initially by the constraints for input rule #4 (constraint set 5), which allow each target complex to be attacked by only one launch field. If the problem is infeasible, and those constraints are relaxed so that missiles from up to two launch fields may attack a single target complex (see constraint set 5), then a new solution is obtained. If this one is not balanced, constraints may be added to require the solution to be balanced.

One way to do this is to first identify which target complex required two launch fields to attack all of its targets. Call this complex j^*. Then require that any launch field that attacks j^* to attack no more than 60 percent of the total number of targets at j^*. This may be enforced by adding a constraint for each launch field that says the total number of missiles of any type from this launch field to j^* attacking any type of target must be less than or equal to 0.6 times the total number of targets at j^*. Mathematically, this becomes:

$$\sum_{k=1}^{MT}\sum_{l=1}^{TT} X_{ij^*kl} \leq 0.6 \times \sum_{l=1}^{TT} n_{j^*l} \quad \text{for every } i.$$

After this set of constraints has been added, the problem may be re-solved. If there is a feasible solution to this formulation, it will be balanced since only one target complex may be attacked by two launch fields and this constraint forces j^* to be that one. However, the resulting solution may not be as good in terms of minimum flight time or minimum duration as the previous solution. This is true because there may be a better feasible, balanced solution with a different target complex being attacked by two launch fields.

The only way to know if such a solution exists is to force a solution with a different target complex being attacked by two launch fields. One way to force this is to remove the above added constraints and add one that requires j^* to be attacked by only one launch field. This constraint may be expressed in terms of the S_{ij} variables:

$$\sum_{i=1}^{LF} S_{ij^*} = 1.$$

This constraint forces target complex j^* to be attacked by only one launch field, but it does not guarantee anything else about the solution. The solution obtained after adding this constraint may be both feasible and balanced and better than the balanced solution where j^* is attacked by two launch fields, or it may turn out to be infeasible, or the solution obtained may be worse (longer flight time or duration), or this solution may be unbalanced as well. The process of identifying the best feasible, balanced solution may be time consuming.

(ii) Input rule #15 says that during each wave, every target in a given complex must be hit at the same time. This is ensured by constraint set 6.

3.3.3 The Complete Model. The entire mixed integer formulation of the problem is

$$min\ W_1O_1 + W_2O_2$$

Subject To:

$$\begin{aligned}
\textstyle\sum_{i=1}^{LF}\sum_{k=1}^{MT} X_{ijkl} &= n_{jl} && \text{for every } j, l \\
\textstyle\sum_{j=1}^{TC}\sum_{l=1}^{TT} X_{ijkl} &\le m_{ik} && \text{for every } i, k
\end{aligned}$$

$$\left.\begin{aligned}
Y_{ijk} &\le \textstyle\sum_{l=1}^{TT} X_{ijkl} \\
Y_{ijk} &\ge \frac{\sum_{l=1}^{TT} X_{ijkl}}{\sum_{l=1}^{TT} n_{ijl}}
\end{aligned}\right\}\ \text{for every } i, j, k$$

$$\left.\begin{aligned}
S_{ij} &\le \textstyle\sum_{k=1}^{MT} Y_{ijk} \\
S_{ij} &\ge \frac{1}{MT}\textstyle\sum_{k=1}^{MT} Y_{ijk}
\end{aligned}\right\}\ \text{for every } i, j$$

$$\begin{aligned}
\textstyle\sum_{i=1}^{LF} S_{ij} &= 1 && \text{for every } j \\
Z_j &\ge T_{ijk}Y_{ijk} && \text{for every } i, j, k \\
T_{first} &\ge Z_j + \delta_j(-T_{max}) && \text{for every } j \\
\textstyle\sum_{j=1}^{TC} \delta_j &= TC - 1 && \\
T_{first} &\le Z_j && \text{for every } j \\
T_{last} &\ge Z_j && \text{for every } j \\
Z_{j^*} &\le Z_{j'} && \text{for every } j^* \in j_G, j' \in j_F \\
Z_{j'} &\le Z_{j^\dagger} && \text{for every } j' \in j_F, j^\dagger \in j_P
\end{aligned}$$

$$\begin{aligned}
Dur &= T_{last} - T_{first} \\
T_{first} - O_1 &= T_{min} \\
Dur - O_2 &= 0 \\
Dur &\le Pct \times T_{first}
\end{aligned}$$

Variable Types:

$$\begin{aligned}
X_{ijkl} &\ge 0 \text{ for every } i, j, k, l \\
Y_{ijk} &= 0 \text{ or } 1 \text{ for every } i, j, k \\
S_{ij} &= 0 \text{ or } 1 \text{ for every } i, j
\end{aligned}$$

$$\begin{aligned}
Z_j &\geq 0 \text{ for every } j \\
\delta_j &= 0 \text{ or } 1 \text{ for every } j \\
T_{first} &\geq 0 \\
T_{last} &\geq 0 \\
Dur &\geq 0 \\
O_1 &\geq 0 \\
O_2 &\geq 0
\end{aligned}$$

Since X_{ijkl} is the number of missiles at launch field i allocated to target complex j which are of type k and attack type l targets, each X_{ijkl} should be an integer variable. Although X_{ijkl} is not required to be integer-valued in this formulation of the model, each assume integer value when every n_{jl} is an integer and when each target complex is attacked by only one launch field and only one missile type. If this is not the case, X_{ijkl} could have non-integer value in the optimal solution.

Since each n_{jl} used in the model is an integer, the only way this could happen is if more than one launch field is attacking the target complex or if a launch field is using more than one missile type to attack a target complex. If the latter case results in two X_{ijkl}s having non-integer value, simply round one of them down and round the other one up. This results in a feasible solution since there must be an integer number of missiles of each type available at each launch field. If there are two launch fields striking the same target complex and the associated X_{ijkl}s are non-integer, then round the larger X_{ijkl} down and the smaller one up. This is again feasible because there must be an integer number of missiles of each type available at each launch field. This also helps ensure the allocation is balanced [see the handling of input rule #14, section 3.3.2, item 12(m), p. 29–30].

If each X_{ijkl} is required to have integer value, the model has an additional $LF \times TC \times MT \times TT$ integer variables. This could cause a significant increase in the solution time.

3.4 Solution Methodology

The mathematical formulation presented in section 3.3.3 is a mixed-integer program (MIP) because it has both integer and continuous variables. The integer variables in the above formulation are all binary. Thus, a MIP solver is needed to solve this problem. However, the solver needs the problem data entered into the above formulation before it can solve the problem. Since J-8 desired a FORTRAN routine that they could incorporate into SINBAC, a FORTRAN computer program was written to read in the problem data, transform this data for use in the above formulation, and then transmit the formulated problem to the solver. The program also takes the solution produced by the solver and provides it in an easily understood format.

3.4.1 Procedure. The following is an outline of a procedure which may be used to solve J-8's problem. Figures 1 and 2 present a flow diagram of the procedure.

STEP 1: Read in all the data.

a. Read in the number of launch fields (LF), number of target complexes (TC), number of missile types (MT), number of target types (TT), the percent of the minimum flight time that the duration can be (Pct), the percent backup required at each launch field ($Pctbu$), and the relative importance of minimizing the flight time and the duration of the attack (W_1 and W_2 respectively).

b. Read in the flight time data (T_{ijk}), number of missiles by type at each launch field (m_{ik}), and number of targets by type at each target complex (n_{jl}).

STEP 2: Formulate the model presented in section 3.3.3 using the data collected in STEP 1.

STEP 3: Solve the problem using the Zero/One Optimization Methods (ZOOM).

STEP 4: Write the solution to a file. If the user is satisfied with the solution, then go to STEP 5. If the user desires to modify the solution directly, then perform the modifications and go to STEP 5. If the user wants to modify the problem and re-solve it, then make the modifications and go to STEP 3.

STEP 5: Process and print the final solution.

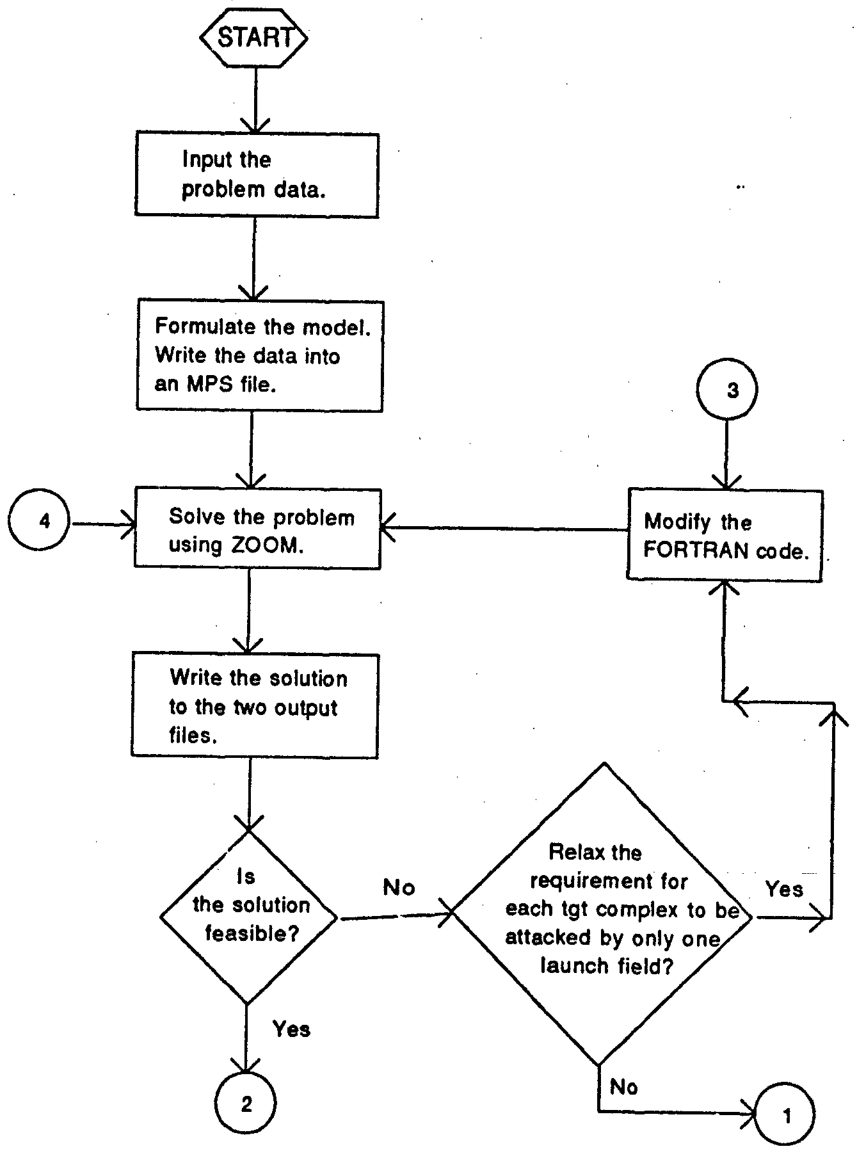
START
Input the problem data.
Formulate the model. Write the data into an MPS file.
Solve the problem using ZOOM.
Write the solution to the two output files.
Is the solution feasible?
Yes
No
Relax the requirement for each tgt complex to be attacked by only one launch field?
Yes
No
Modify the FORTRAN code.
1
2
3
4

Figure 1. Flow Diagram

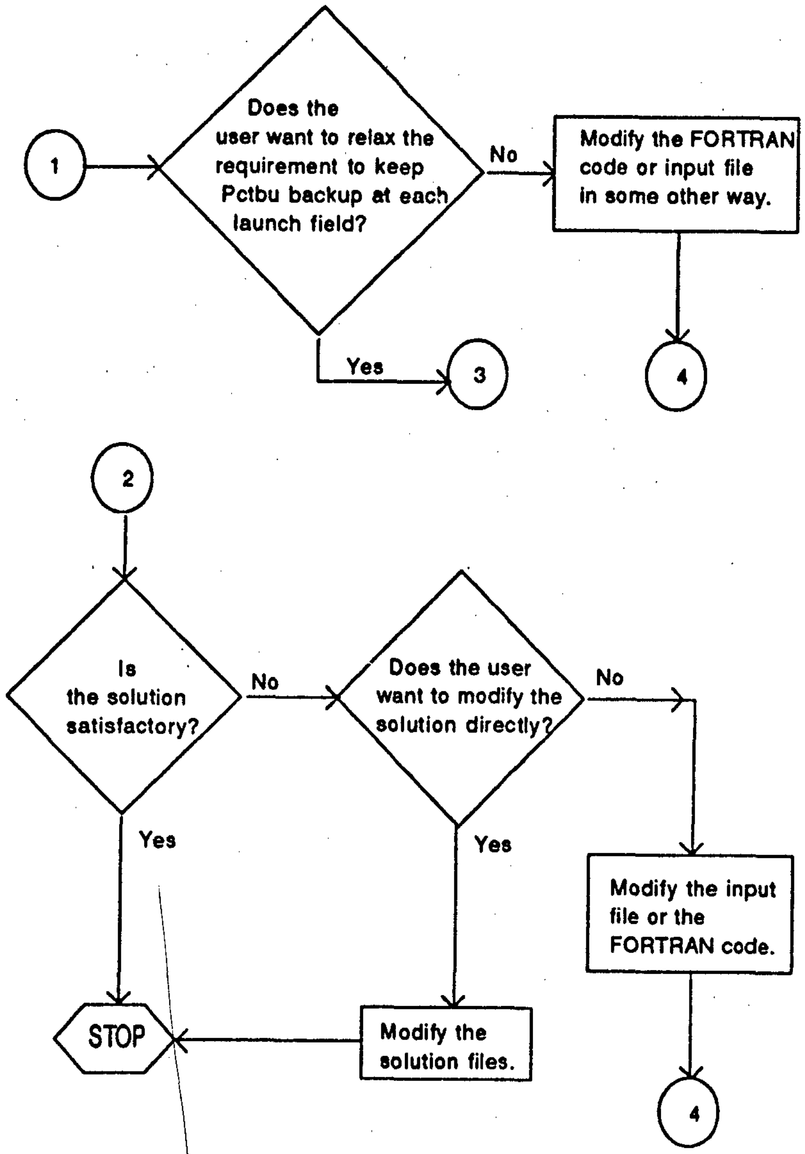
1
Does the user want to relax the requirement to keep Pctbu backup at each launch field?
No
Modify the FORTRAN code or input file in some other way.
Yes
3
4
2
Is the solution satisfactory?
No
Does the user want to modify the solution directly?
No
Yes
Yes
Modify the input file or the FORTRAN code.
STOP
Modify the solution files.
4

Figure 2. Flow Diagram (Continued)

IV. Results and Findings

4.1 Introduction

Chapter III presented a mathematical formulation of the J-8 missile allocation problem as a mixed-integer program. In this chapter, the results and findings of the procedure presented in Chapter III are discussed. The presentation begins with a more detailed description of the procedure.

To accomplish Steps 1 and 2 of the procedure (see Section 3.4.1), a FORTRAN computer program (see Appendix B) was written that receives as input the data for the problem. This data is then put into the mixed-integer formulation as specified in Chapter III and the problem is written to a Mathematical Programming System (MPS) formatted file. At Step 3, the Zero/One Optimization Methods (ZOOM) routines read the MPS file and solve the missile allocation problem. At Step 4, the solution produced by ZOOM is sent to two output files: one that is in the format J-8 wants so that they can review the solution, and one that is used to store the allocation in the format that SINBAC needs.

If the problem is infeasible, the user may modify either the input data or the FORTRAN code. If the user chooses to modify the code, one or more of the following sets of constraints could be relaxed:

1. Allow a target complex to be attacked by two launch fields, so change

$$\sum_{i=1}^{LF} S_{ij} = 1 \quad \text{for each j}$$

to

$$\sum_{i=1}^{LF} S_{ij} \leq 2 \quad \text{for each j},$$

add the constraint

$$\sum_{i=1}^{LF}\sum_{j=1}^{TC} S_{ij} \leq TC + 1,$$

and re-solve;

2. Do not require that targets attacked by fair missiles in the initial wave be attacked by missiles from the same launch field in the second wave. This constraint may be relaxed by not dividing m_{ik} for fair missiles in half prior to solving (see the handling of input rule #8, section 3.3.2, item 12(g), p. 27);
3. Do not require each launch field to have a *Pctbu* backup, so do not multiply each m_{ik} by $(1 - Pctbu)$, but instead add the following constraint:

$$\sum_{i=1}^{LF}\sum_{j=1}^{TC}\sum_{k=1}^{MT}\sum_{l=1}^{TT} X_{ijkl} \leq (1 - Pctbu) \times \sum_{i=1}^{LF}\sum_{k=1}^{MT} m_{ik}$$

which ensures that no more than $(1 - Pctbu)$ percent of the total number of available missiles are used in the allocation.

If the solution is unacceptable to the user, either modify the solution files directly, modify the input data file and re-solve the problem, or modify the FORTRAN routines directly and re-solve the problem.

4.2 Solution to the Unclassified Problem

The FORTRAN program (with ZOOM) was used to solve the unclassified problem. Using 10 percent of the earliest flight time for the limit as to how long the duration could be $(Pct = 0.1)$, using weighting factors $W_1 = 10$ and $W_2 = 1$ (assuming it is much more important to minimize the flight time of the first missile than it is to minimize the duration of the attack), and requiring each launch field to maintain a 10 percent backup $(Pctbu = 0.1)$, the following solution was obtained:

Objective Function Value = 22.575

$$\begin{array}{llllll} T_{first} & = 34.027 & T_{last} & = 37.43 & Dur & = 3.403 \\ O_1 & = 1.917 & O_2 & = 3.403 & & \end{array}$$

$$\begin{array}{llllll}
X_{1211} = 3 & & X_{1212} = 18 & & X_{1221} = 15 \\
X_{2321} = 21 & & X_{3422} = 12 & & X_{3623} = 2 \\
X_{4111} = 5 & & X_{4121} = 14 & & X_{4511} = 24 \\
\\
Y_{121} = 1 & & Y_{122} = 1 & & Y_{232} = 1 \\
Y_{342} = 1 & & Y_{362} = 1 & & Y_{411} = 1 \\
Y_{412} = 1 & & Y_{451} = 1 & & \\
\\
S_{12} = 1 & & S_{23} = 1 & & S_{34} = 1 \\
S_{36} = 1 & & S_{41} = 1 & & S_{45} = 1 \\
\\
Z_1 = 35.15 & & Z_2 = 35.98 & & Z_3 = 34.027 \\
Z_4 = 37.43 & & Z_5 = 37.43 & & Z_6 = 37.43 \\
\\
\delta_1 = 1 & & \delta_2 = 1 & & \delta_4 = 1 \\
\delta_5 = 1 & & \delta_6 = 1 & &
\end{array}$$

$$\text{all other variables } = 0.$$

4.2.1 Allocation Summary and Explanation. Table 1 summarizes the allocation specified by the solution to the unclassified problem.

This solution means that:

1. Since $X_{1211} = 3$ and $X_{1212} = 18$, New Bern launches three good missiles at good targets at Atlanta/Tybee and 18 good missiles at fair targets at Atlanta/Tybee. Also, $Z_2 = 35.98$, so these missiles impact at time 35.98, which is the actual flight time of good missiles from New Bern to Atlanta/Tybee. New Bern started with a total of 24 good missiles, so it is left with three good missiles as backup.

Table 1. Sample Problem Allocation Summary

		TARGET COMPLEX TARGET TYPE						
LAUNCH	MISSILE	Macon	Atl/Tyb		Savannah	Columbus	Brunswick	Athens
FIELD	TYPE	good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair				21			
Raleigh	fair					12		2
Wilmington	good	5					24	
	fair	14						
Charlotte								
Lumberton								

2. $X_{1221} = 15$, which means New Bern also launches 15 fair missiles at good targets at Atlanta/Tybee. Since $Z_2 = 35.98$, these missiles are launched in order to impact at 35.98, so their launch time is delayed by 2.74 minutes (the actual flight time of fair missiles from New Bern to Atlanta/Tybee is 33.24 minutes). This delay should occur so that all missiles launched at Atlanta/Tybee impact at the same time. Also, in the second wave, New Bern should launch 15 fair missiles at Atlanta/Tybee. These missiles should be launched at time 32.74 (30 minutes after the first wave is launched) so that they detonate 30 minutes after the ones in the first wave. Since New Bern started with a total of 34 fair missiles, this leaves it with four fair missiles as backup.

3. Since $X_{2321} = 21$, Durham launches 21 fair missiles at the 21 good targets at Savannah. Also, $Z_3 = 34.027$, so these missiles should be launched in order to impact at time 34.027. This means that their launch should be delayed by 1.577 minutes (the actual flight time of fair missiles from Durham to Savannah is 32.45 minutes). This delay should occur in order to keep the duration of the attack within the given percentage ($Pct = 0.1$) of the earliest flight time. During the second wave, Durham also launches 21 fair missiles at Savannah.

These second wave missiles should be launched at time 31.577. Durham started with 51 fair missiles, so this leaves it with nine fair missiles as backup.

4. $X_{3422} = 12$, so Raleigh launches 12 fair missiles at the 12 fair targets at Columbus. Since $Z_4 = 37.43$, these missiles should be launched in order to impact at time 37.43, so their launch must be delayed by 4.01 minutes (the actual flight time of fair missiles from Raleigh to Columbus is 33.42 minutes). This delay should occur so that all good targets are attacked no later than each fair target. Also, $X_{3623} = 2$ means that Raleigh launches two fair missiles at the two poor targets at Athens. Since $Z_6 = 37.43$, these missiles should be launched in order to impact at time 37.43, so their launch must be delayed by 2.09 minutes (the actual flight time of fair missiles from Raleigh to Athens is 35.34 minutes). Again, this delay should occur so that good targets are attacked at least as early as fair targets. During the second wave, Raleigh also launches 12 fair missiles at Columbus and two fair missiles at Athens. These missiles should launch at times 34.01 and 32.09, respectively, in order to impact 30 minutes after the first wave missiles impact. Raleigh started with 34 fair missiles, so this leaves it with six fair missiles as backup.

5. Since $X_{4111} = 5$, Wilmington should launch five good missiles at Macon. $Z_1 = 35.15$, so these missiles should be launched in order to impact at time 35.15, which is the actual flight time of good missiles from Wilmington to Macon. Also, $X_{4511} = 24$, so Wilmington launches 24 good missiles at the 24 good targets at Brunswick. Since $Z_5 = 37.43$, these missiles should be launched in order to impact at time 37.43, which is the actual flight time of good missiles from Wilmington to Brunswick. Wilmington started with 33 good missiles, so this leaves it with four good missiles as backup.

6. $X_{4121} = 14$ means that Wilmington launches 14 fair missiles at good targets at Macon. Since $Z_1 = 35.15$, these missiles should also be launched in order to impact at time 35.15, so their launch must be delayed by 2.85 minutes (the

actual flight time of fair missiles from Wilmington to Macon is 32.30 minutes). This delay should occur so that all missiles attacking Macon impact at the same time. During the second wave, Wilmington should launch 14 additional fair missiles at the same targets at Macon. These missiles should be launched at time 32.85 in order to impact 30 minutes after the first wave does. Wilmington started with 38 fair missiles, so this leaves it with ten fair missiles as backup.

7. Since Charlotte and Lumberton were designated as backup launch fields, they do not launch any missiles, so they have all of their missiles as backup.

This solution is summarized by the output files in the formats that J-8 requested (see Appendix C). This solution was obtained in 57.15 minutes of CPU (Central Processing Unit) time on a VAX/VMS 6420 computer system. Obviously, a larger problem may take longer to solve. Fortunately, the actual problem will probably decrease in size in the future.

4.3 Model Verification and Validation

4.3.1 Comparison to GAMS/ZOOM. In order to verify that the model presented in Chapter III—and implemented by the FORTRAN program—solved the problem properly, this problem was also input into GAMS (the General Algebraic Modeling System). GAMS solves mixed-integer problems with a modified version of ZOOM which they call GAMS/ZOOM. The solution found by GAMS/ZOOM is as follows (see Appendix D for the GAMS input file):

Objective function value = 22.575

$$\begin{array}{lclclcl} T_{first} & = & 34.027 & T_{last} & = & 37.43 & Dur = 3.403 \\ O_1 & = & 1.917 & O_2 & = & 3.403 & \end{array}$$

$$\begin{array}{lll}
X_{1211} = 3 & X_{1212} = 18 & X_{1221} = 15 \\
X_{2121} = 19 & X_{2623} = 2 & X_{3422} = 12 \\
X_{4311} = 6 & X_{4321} = 15 & X_{4511} = 23 \\
X_{4521} = 1 & &
\end{array}$$

$$\begin{array}{lll}
Y_{121} = 1 & Y_{122} = 1 & Y_{212} = 1 \\
Y_{262} = 1 & Y_{342} = 1 & Y_{431} = 1 \\
Y_{432} = 1 & Y_{451} = 1 & Y_{452} = 1
\end{array}$$

$$\begin{array}{lll}
S_{12} = 1 & S_{21} = 1 & S_{26} = 1 \\
S_{34} = 1 & S_{43} = 1 & S_{45} = 1
\end{array}$$

$$\begin{array}{lll}
Z_1 = 34.027 & Z_2 = 37.43 & Z_3 = 37.43 \\
Z_4 = 37.43 & Z_5 = 37.43 & Z_6 = 37.43
\end{array}$$

$$\begin{array}{lll}
\delta_2 = 1 & \delta_3 = 1 & \delta_4 = 1 \\
\delta_5 = 1 & \delta_6 = 1 &
\end{array}$$

all other variables $= 0$

4.3.1.1 Allocation Summary and Explanation. Table 2 summarizes the allocation specified by the GAMS/ZOOM solution to the unclassified problem.

The solution obtained by GAMS/ZOOM is clearly different from the one obtained by ZOOM. In particular, flight times to various complexes differ. This is the result of a different allocation. For example, Macon is now attacked by Durham rather than Wilmington. However, they are the same in terms of the objective function value (to within the ZOOM and GAMS specified tolerance of 0.1). This means that there could be alternate optimal solutions for any given problem. There is a

Table 2. Allocation Summary for the GAMS/ZOOM Solution

		TARGET COMPLEX TARGET TYPE						
LAUNCH	MISSILE	Macon	Atl/Tyb		Savannah	Columbus	Brunswick	Athens
FIELD	TYPE	good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair	19						2
Raleigh	fair					12		
Wilmington	good				6		23	
	fair				15		1	
Charlotte								
Lumberton								

slightly different missile allocation specified by each approach, but the flight time of the first missile (T_{first}) and the duration of the attack (Dur) are the same.

4.3.2 Face Validation. This model and the results of solving the unclassified sample problem were shown to J-8 representatives. Each of them believed that the solution obtained by the model was a valid optimal solution to the problem. Based upon expert opinion, the solutions obtained using this model are valid solutions to the J-8 missile allocation problem.

4.4 Solution Using Number of Warheads

It is possible that sometime in the future J-8 could designate missiles for this problem that do not have 10 warheads each. In fact, each missile type could have a unique number of warheads. Therefore, this problem was re-solved with ZOOM using the same unclassified data, but using the number of warheads at each launch field and the number of targets (not divided by 10) at each target complex. The results are

Objective function value = 22.575

T_{first}	=	34.027	T_{last}	=	37.43	Dur	=	3.403
O_1	=	1.917	O_2	=	3.403			
X_{1211}	=	41	X_{1212}	=	175	X_{1221}	=	134
X_{2121}	=	185	X_{2623}	=	20	X_{3422}	=	120
X_{4311}	=	210	X_{4511}	=	87	X_{4521}	=	153
Y_{121}	=	1	Y_{122}	=	1	Y_{212}	=	1
Y_{262}	=	1	Y_{342}	=	1	Y_{431}	=	1
Y_{451}	=	1	Y_{452}	=	1			
S_{12}	=	1	S_{21}	=	1	S_{26}	=	1
S_{34}	=	1	S_{43}	=	1	S_{45}	=	1
Z_1	=	34.027	Z_2	=	35.98	Z_3	=	35.60
Z_4	=	37.43	Z_5	=	37.43	Z_6	=	37.43
δ_2	=	1	δ_3	=	1	δ_4	=	1
δ_5	=	1	δ_6	=	1			

all other variables = 0

4.4.1 Allocation Summary and Explanation. Table 3 summarizes the allocation specified by the solution to the unclassified problem when the number of warheads at each launch field is used instead of the number of missiles.

The measure of performance is the same for this problem as it was for the original problem. In other words, this formulation yields the same solution in terms

Table 3. Allocation Summary Using Warheads Instead of Missiles

		TARGET COMPLEX TARGET TYPE						
LAUNCH	MISSILE	Macon	Atl/Tyb		Savannah	Columbus	Brunswick	Athens
FIELD	TYPE	good	good	fair	good	fair	good	poor
New Bern	good		41	175				
	fair		134					
Durham	fair	185						20
Raleigh	fair					120		
Wilmington	good				210		87	
	fair						153	
Charlotte								
Lumberton								

of the flight time of the missile that impacts first and the duration of the attack. However, the allocation is slightly different.

Since $X_{1211} = 41$, this would indicate that New Bern should launch 41 good warheads at Atlanta/Tybee. However, since each good missile carries 10 warheads, this would mean that New Bern would be launching 4.1 good missiles. This is impossible, so New Bern would really launch either four or five good missiles at Atlanta/Tybee. A closer examination of the solution reveals that New Bern should also launch 134 fair warheads at Atlanta/Tybee (since $X_{1221} = 134$), which would mean a launch of 13 or 14 fair missiles. Since the number of warheads carried by both good and fair missiles is 10, each launch field started with a multiple of 10 warheads available for the attack (not reserved for backup). Therefore, New Bern started with at least 50 good warheads (five good missiles) and at least 140 fair warheads (14 fair missiles) available for the attack. This means that in the actual allocation, New Bern would either be scheduled to launch five good and 13 fair missiles or four good and 14 fair missiles at Atlanta/Tybee. Each of these options is feasible, so it would be the decision maker's choice as to which allocation would be used.

A similar result was obtained in the allocation of warheads from Wilmington to Brunswick ($X_{4511} = 87$ and $X_{4521} = 153$). Here the decision maker would have to choose between launching eight good and 16 fair missiles, or nine good and 15 fair missiles from Wilmington to Brunswick.

Sample output files generated by the FORTRAN program are available in Appendix C.

4.5 *Parametric Analysis*

In order to test the sensitivity of the unclassified sample problem to changes in the input parameters, the parameters Pct, W_1, and W_2 were alternately varied and the problem re-solved. The parameter $Pctbu$ was not varied since it is currently fixed at 10 percent.

Nine different allocations were performed using varying values for Pct, W_1, and W_2. Table 4 summarizes the pertinent results of the nine different solutions.

Table 4. Parametric Analysis Summary

CASE	Pct	W_1	W_2	T_{first}	DUR	CPU TIME
1	0.1	10	1	34.027	3.403	0:57.15
2	0.05	10	1	35.648	1.782	1:39.13
3	0.15	10	1	32.548	4.882	0:53.71
4	0.1	2	1	34.027	3.403	1:39.45
5	0.1	1	2	37.43	0	1:42.58
6	0.05	2	1	35.648	1.782	1:37.45
7	0.05	1	2	37.43	0	1:39.94
8	0.1	1	1	34.027	3.403	1:46.52
9	0.05	1	1	35.648	1.782	1:39.08

Case 1 is the original choices of the input parameters with the full results summarized in section 4.2. All other cases have their complete solutions and allocation summaries listed in Appendix E. The CPU times for each of these cases were obtained on a VAX/VMS 6420 and are in hours:minutes (to the nearest one-hundredth of a minute).

Case 2 is notable for its interesting results. Pct was set at 0.05 (Dur was limited to five percent of T_{first}) and it resulted in the same value for T_{last} with T_{first} increased so that the percentage restriction on the duration of the attack was still met.

The result from case 3 seems to indicate that for a given set of problem data, as the duration is allowed to increase (Pct gets larger), the flight time of the first missile (T_{first}) decreases while the flight time of the last missile (T_{last}) stays the same.

The result of case 1 compared with the result of case 4, and the result of case 2 compared with 6, seems to indicate that the optimal solution is the same when W_1 is greater than W_2 (i.e., minimizing the flight time of the missile that impacts first is more important than minimizing the duration of the attack) for a fixed duration. Similarly, the results of cases 5 and 7 seem to indicate that if W_2 is greater than W_1 (it is more important to minimize the duration), then the value of T_{first} is set to equal T_{last} so that the duration is zero.

V. Conclusions and Recommendations

5.1 Conclusions

The solution to the unclassified sample problem obtained by the model presented in Chapter III was compared to the solution obtained by GAMS/ZOOM. Since the measure of performance was the same, the solution was verified. Also, expert opinion was consulted and the solution obtained was considered valid. Therefore, the model presented in Chapter III may be solved to derive optimal solutions to missile allocation problems.

This model is flexible because it allows for various numbers of launch fields, target complexes, missile types, and target types. This model also allows the decision maker to specify the relative importance of each of the two objectives: minimize the first impact time and minimize the duration of the attack. This flexibility makes the model usable for a variety of scenarios.

However, this model would not be practical to use if the number of binary variables becomes significantly large. The number of binary variables increases by $(MT \times TC) + TC$ if the number of launch fields increases by one. Similarly, if the number of target complexes or missile types increases by one, the number of binary variables increases by $(LF \times MT) + LF + 1$ or $LF \times TC$ respectively. Since the solution times could potentially increase exponentially as the number of binary variables increases, the model may become impractical to use once the number of binary variables exceeds around 100 or so. However, in the foreseeable future, the numbers of launch fields and target complexes should decrease, and the number of missile types could drop to one, so this model should be usable for at least several years.

Of course, the future is unknown. Input rules could change; new ones could be added or current ones deleted. These changes to the input rules could require the model to be modified. It seems fairly simple to accommodate foreseeable changes by changing, adding, or deleting constraints as appropriate. Therefore, even if changes in the world situation dictate significant changes to the model as presented here, it should still serve as a solid basis for building a model which would represent the new situation.

5.2 Model Limitations

This model is a good representation of the J-8 missile allocation problem, but it is not perfect. There is an input rule that is not modeled explicitly. Input rule #14 says that no "strays" are allowed. This means that the solution should be somewhat balanced (see the handling of input rule #14, section 3.3.2, item 12(m), p. 29). If there is a feasible solution that has each target complex attacked by only one launch field, this requirement will be met. However, if there is no such solution, then any target complex attacked by more than one launch field may have an unbalanced allocation (i.e., it may have one launch field attacking 90 percent of its targets and another attacking 10 percent). There is nothing in the model to prevent this from happening. In Chapter III, one approach that may be tried was discussed (see section 3.3.2, item 12(m), p. 29–30). However, this approach is not guaranteed to yield the optimal solution. In fact, it is not guaranteed to yield even a feasible solution. This is because it is theoretically possible to have a set of data that has no balanced solutions that are feasible.

When initially modeling the J-8 missile allocation problem, the constraints that require the flight time of the missile that impacts first (T_{first}) to be less than or equal to the time of attack of each target complex (Z_j) were not included. It was observed that some solutions that were generated had a first impact time that was greater than one or more of the target complex attack times ($T_{first} > Z_j$ for at least one j). This solution is not consistent with the definition of Z_j, which is the time of attack of target complex j. Therefore, the constraints mentioned above were added. This had the effect of producing solutions that are consistent with the variable definitions, but the solution times were observed to increase. The solution times obtained for the unclassified sample problem before these constraints were added were in the neighborhood of 25 to 30 minutes. After these constraints were added, solution times jumped to between 55 and 60 minutes. This is an increase by a factor of two. For this problem, this does not cause the solution times to become prohibitive. However, for a larger problem, including this set of constraints may result in the solution time becoming too large for the model to be used practically. If this is the case, dropping these constraints may result in the solution time becoming acceptable. If these constraints are dropped, the launch times of the missiles allocated to impact prior to T_{first} would have to be manually delayed in order to meet all of the constraints.

5.3 Recommendations

Since the input rule for requiring a balanced solution is not explicitly modeled, further research should concentrate on finding a way to model this requirement so as to guarantee an optimal solution that is both balanced and feasible.

This model generates optimal allocations based on the decision maker's inputs for the weighting factors for the two objectives (W_1 and W_2) and the fraction of the first impact time that the duration is allowed to be (Pct). Although parametric analysis was conducted and the results reported in Chapter IV, further research may reveal insights as to potentially "good" values for these parameters based on the decision maker's desires.

Since the solution time for solving a given problem is heavily dependent on the number of binary variables, every effort should be made to eliminate ones that are unnecessary. For instance, the model calls for a binary variable (Y_{ijk}) for each launch field/target complex/missile type combination. However, since most of the launch fields do not have all of the missile types, not all the Y_{ijk}s are necessary. For the unclassified sample problem, the basic model calls for $6 \times 6 \times 2 = 72$ different Y_{ijk} variables. However, only three of the launch fields (New Bern, Wilmington, and Charlotte) have both missile types, so the number of Y_{ijk} variables can be reduced to $(3 \times 6 \times 2) + (3 \times 6 \times 1) = 54$. In addition, two of the launch fields are reserved for backup only, so the number of necessary Y_{ijk} variables may be reduced to $(2 \times 6 \times 2) + (2 \times 6 \times 1) = 36$. If all such reductions are made, the number of necessary binary variables in the unclassified sample problem may be reduced from 114 to 66.

Appendix A. Sample Problem–Unclassified Data

A.1 Launch Fields and Missile Data

There are six launch fields: New Bern, Durham, Raleigh, Wilmington, Charlotte, and Lumberton. Each launch field has one or both missile types as follows:

Table 5. Launch Fields and Missile Data

LAUNCH FIELD	TYPE	MISSILES	WARHEADS
New Bern	good	24	240
	fair	34	340
Durham	fair	51	510
Raleigh	fair	34	340
Wilmington	good	33	330
	fair	38	380
Charlotte	good	24	240
	fair	25	250
Lumberton	fair	58	580

A.2 Target Complexes and Target Data

There are seven target complexes: Macon, Atlanta, Savannah, Columbus, Brunswick, Tybee, and Athens. Each target complex has either good, fair, or poor targets. Table 6 contains this data.

Table 6. Target Complexes and Target Data

TARGET COMPLEX	TYPE	NUMBER
Macon	good	185
Atlanta	good	175
Savannah	good	210
Columbus	fair	120
Brunswick	good	240
Tybee	fair	175
Athens	poor	20

A.3 Flight Time Data

Table 7 lists the flight time from each launch field to each target complex by missile type.

Table 7. Flight Time Data

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX						
		Macon	Atlanta	Savannah	Columbus	Brunswick	Tybee	Athens
New Bern	good	34.16	34.81	33.89	35.33	36.49	37.15	38.22
	fair	32.61	32.76	32.66	33.10	33.61	33.72	34.82
Durham	fair	32.11	32.03	32.45	32.51	32.96	32.78	34.71
Raleigh	fair	32.95	33.02	33.09	33.42	33.92	33.93	35.34
Wilmington	good	35.15	35.26	35.60	36.26	37.43	34.42	40.78
	fair	32.30	32.20	32.67	32.70	33.14	32.93	34.94
Charlotte	good	35.45	35.66	35.76	36.58	37.76	37.87	37.81
	fair	32.58	32.52	32.90	33.00	33.46	33.30	35.19
Lumberton	fair	33.41	33.44	33.60	33.88	34.38	34.33	35.91

Appendix B. FORTRAN Code

B.1 Introduction

The program given in Appendix B was written in its entirety by the author of this thesis. This program calls ZOMAIN, which is the top level routine for ZOOM, as a subroutine. ZOMAIN and all other subroutines contained in ZOOM were written by Dr. Roy E. Marsten, currently of the Georgia Institute of Technology.

In order to get ZOOM to interact with the FORTRAN program that this author wrote, and to get it to run on the VAX/VMS system at the Air Force Institute of Technology, the following changes were made to the ZOOM code:

1. ZOMAIN was changed from a PROGRAM to a SUBROUTINE.
2. All references to the subroutine TIMER were commented out.
3. In the subroutine ZDRIVE, a variable was initialized to one instead of zero because that value is used as an array dimension.
4. The parameter MAXVAR and the array VARVAL(MAXVAR) are passed as arguments into ZOMAIN, then ZDRIVE, and then MPSOUT in order to collect the values of the solution variables to pass back to the main program.
5. Statements were added to various subroutines to print error messages to the file SILOATT.ERR whenever the program encounters difficulties.

B.2 Limitations

The FORTRAN program written for this thesis implements this model and solves missile allocation problems. However, it has some limitations:

1. The subroutine that reads in the data is "fragile" in that if the user inputs the wrong type of data (i.e., entering integer data where real data is required), the program terminates. Future research should include modifying this subroutine

so that if the user accidentally enters the wrong type of data, the program does not terminate but instead gives an error message as to what the problem is.

2. If the user wants to modify a constraint, the FORTRAN code must be modified in each part of the program that applies to that constraint, particularly in the formulation subroutine (FORMUL) which formulates the data into the appropriate mixed-integer program and writes it to an MPS file. Future research should concentrate on making it easier to modify constraints. This could include having a main menu that would allow the user to choose inputting a new problem, modifying an existing input data file, modifying the current solution files directly, or modifying constraints on the problem.

B.3 FORTRAN Code

```
      PROGRAM SILOATT
C**********************************************************************************
C
C  NAME:  SILOATT
C
C  REV  DATE OF
C  NO.  CHANGE   DESIGNER      CCR   DESCRIPTION
C  ---  -------  ------------  ----  ----------------------------------------
C  000   9FEB93  R. PACE             ORIGINAL RELEASE
C
C  DESCRIPTION:  DETERMINES THE OPTIMAL MISSILE ALLOCATION FOR THE SILO ATTACK
C
C  CALLED BY:  NONE
C
C  CALLS:
C
C    INPUT   - COLLECTS THE DATA FOR THE SILO ATTACK PLAN
C    PROCESS - PROCESSES THE DATA FOR THE SILO ATTACK PLAN
C    FORMUL  - WRITES THE DATA INTO AN MPS-FORMAT FILE
C    ZOMAIN  - INVOKES ZOOM TO SOLVE THE ALLOCATION
C    PRNTSOL - PRINTS THE SOLUTION INTO A FILE
C
C  CALLING SEQUENCE:  NONE
C
C  FILES:  NONE
C
C  LOCAL VARIABLES:
C
C    BU      - ARRAY STORING ID OF BACKUP LAUNCH FIELDS
```

```
C     LF      - THE NUMBER OF LAUNCH FIELDS
C     LFMT    - THE NUMBER OF LAUNCH FIELD/MISSILE TYPE COMBINATIONS THAT EXIST
C     M       - ARRAY STORING THE NUMBER OF MISSILES OF EACH TYPE AT EACH LAUNCH
C               FIELD
C     MADJ    - ARRAY STORING THE ADJUSTED NUMBER OF MISSILES OF EACH TYPE AT
C               EACH LAUNCH FIELD
C     MT      - THE NUMBER OF MISSILE TYPES
C     N       - ARRAY STORING THE NUMBER OF TARGETS OF EACH TYPE AT EACH TARGET
C               COMPLEX
C     NADJ    - ARRAY STORING THE ADJUSTED NUMBER OF TARGETS OF EACH TYPE AT EACH
C               TARGET COMPLEX
C     NUMBU   - THE NUMBER OF BACKUP LAUNCH FIELDS
C     NUMFAIR - THE NUMBER OF TARGET COMPLEXES THAT CONTAIN NO GOOD TARGETS BUT
C               DO CONTAIN FAIR TARGETS
C     NUMGOOD - THE NUMBER OF TARGET COMPLEXES THAT CONTAIN GOOD TARGETS
C     NUMPOOR - THE NUMBER OF TARGET COMPLEXES THAT CONTAIN NEITHER GOOD NOR
C               FAIR TARGETS BUT DO CONTAIN POOR TARGETS
C     NUMTAR  - ARRAY STORING THE TOTAL NUMBER OF TARGETS AT EACH TARGET COMPLEX
C     PCT     - PERCENT OF THE MINIMUM FLIGHT TIME THAT THE DURATION IS ALLOWED
C               TO BE
C     PCTBU   - THE PERCENT OF THE AVAILABLE MISSILES AT EACH LAUNCH FIELD
C               THAT MUST BE RESERVED FOR BACKUP
C     T       - ARRAY STORING THE FLIGHT TIME OF A TYPE K MISSILE FROM LAUNCH
C               FIELD I TO TARGET COMPLEX J
C     TC      - THE NUMBER OF TARGET COMPLEXES
C     TCTT    - THE NUMBER OF TARGET COMPLEX/TARGET TYPE COMBINATIONS THAT EXIST
C     TMAX    - THE LARGEST POSSIBLE FLIGHT TIME
C     TMIN    - THE SMALLEST POSSIBLE FLIGHT TIME
C     TT      - THE NUMBER OF TARGET TYPES
C     VARVAL  - ARRAY THAT STORES THE SOLUTION VALUES
C     W1      - THE RELATIVE WEIGHT OF MINIMIZING THE EARLIEST FLIGHT TIME
C     W2      - THE RELATIVE WEIGHT OF MINIMIZING THE DURATION OF THE ATTACK
C
C  ERROR MESSAGES:  NONE
C
C*****************************************************************************
C
C DIMENSION PARAMETERS:
C
C   MAXLF  = MAX NUMBER OF LAUNCH FIELDS
C   MAXMT  = MAX NUMBER OF MISSILE TYPES
C   MAXTC  = MAX NUMBER OF TARGET COMPLEXES
C   MAXTT  = MAX NUMBER OF TARGET TYPES
C   MAXVAR = MAX NUMBER OF VARIABLES
C
      INTEGER  MAXLF, MAXMT, MAXTC, MAXTT, MAXVAR

      PARAMETER ( MAXLF = 10, MAXMT = 5, MAXTC = 10, MAXTT = 5,
     &            MAXVAR = 500 )

C  LOCAL VARIABLE DECLARATIONS:
      INTEGER  BU(MAXLF), LF, LFMT, M(MAXLF,MAXMT), MADJ(MAXLF,MAXMT),
     &         MT, N(MAXTC,MAXTT), NADJ(MAXTC,MAXTT), NUMBU, NUMFAIR,
     &         NUMGOOD, NUMPOOR, NUMTAR(MAXTC), TC, TCTT, TT

      REAL     PCT, PCTBU, T(MAXLF,MAXTC,MAXMT), TMAX, TMIN, W1, W2

      DOUBLE PRECISION VARVAL(MAXVAR)

      CHARACTER*8 CPLX(MAXTC), FIELD(MAXLF), NAME

      CHARACTER*4 MTYPE(MAXMT), TTYPE(MAXTT)

C  INPUT COLLECTS THE DATA FOR THE SILO ATTACK PLAN:
      CALL INPUT(CPLX, FIELD, LF, M, MAXLF, MAXMT, MAXTC, MAXTT, MT,
     &           MTYPE, N, PCT, PCTBU, T, TC, TT, TTYPE, W1, W2)

C  PROCESS PROCESSES THE DATA FOR THE SILO ATTACK PLAN:
      CALL PROCESS(BU, CPLX, LF, LFMT, M, MADJ, MAXLF, MAXMT, MAXTC,
     &             MAXTT, MT, N, NADJ, NUMBU, NUMFAIR, NUMGOOD, NUMPOOR,
     &             NUMTAR, PCTBU, T, TC, TCTT, TMAX, TMIN, TT)

C  FORMUL WRITES THE DATA INTO AN MPS-FORMAT FILE:
      CALL FORMUL(BU, LF, LFMT, MADJ, MAXLF, MAXMT, MAXTC, MAXTT, MT,
     &            NADJ, NUMBU, NUMFAIR, NUMGOOD, NUMPOOR, NUMTAR, PCT,
     &            T, TC, TCTT, TMAX, TMIN, TT, W1, W2)

C  ZOMAIN INVOKES ZOOM TO SOLVE THE ALLOCATION:
      CALL ZOMAIN(MAXVAR, VARVAL)

C  PRNTSOL PRINTS THE SOLUTION INTO FILES:
      CALL PRNTSOL(BU, CPLX, FIELD, LF, MADJ, MAXLF, MAXMT, MAXTC,
     &             MAXTT, MAXVAR, MT, MTYPE, NADJ, NUMBU, T, TC, TT,
     &             TTYPE, VARVAL)

      END

C*****************************************************************************
C*****************************************************************************
      SUBROUTINE INPUT(CPLX, FIELD, LF, M, MAXLF, MAXMT, MAXTC, MAXTT,
     &                 MT, MTYPE, N, PCT, PCTBU, T, TC, TT, TTYPE, W1,
     &                 W2)
C****************************************************************************
C
C  NAME:  INPUT
C
C  REV  DATE OF
C  NO.  CHANGE    DESIGNER     CCR  DESCRIPTION
C  ---  -------  ------------ ----  -----------------------------------------
C  000   9FEB93  R. PACE            ORIGINAL RELEASE
C
```

```
C  DESCRIPTION:  COLLECTS THE DATA FOR THE SILO ATTACK PLAN
C
C  CALLED BY:  SILOATT
C
C  CALLS:  NONE
C
C  CALLING SEQUENCE:
C
CIN  MAXLF  - MAX NUMBER OF LAUNCH FIELDS
C    MAXMT  - MAX NUMBER OF MISSILE TYPES
C    MAXTC  - MAX NUMBER OF TARGET COMPLEXES
C    MAXTT  - MAX NUMBER OF TARGET TYPES
COUT CPLX  - ARRAY STORING THE NAMES OF THE TARGET COMPLEXES
C    FIELD - ARRAY STORING THE NAMES OF THE LAUNCH FIELDS
C    LF    - NUMBER OF LAUNCH FIELDS
C    M     - ARRAY STORING NUMBER OF MISSILES BY TYPE AT EACH LAUNCH FIELD
C    MT    - NUMBER OF MISSILE TYPES
C    MTYPE - ARRAY STORING THE NAMES OF THE MISSILE TYPES
C    N     - ARRAY STORING NUMBER OF TARGETS BY TYPE AT EACH TARGET COMPLEX
C    PCT   - PERCENT OF TFIRST THAT DUR IS ALLOWED TO BE
C    PCTBU - THE PERCENT OF THE AVAILABLE MISSILES AT EACH LAUNCH FIELD
C            THAT MUST BE RESERVED FOR BACKUP
C    T     - ARRAY STORING FLIGHT TIMES OF MISSILES (BY TYPE) FROM EACH LAUNCH
C            FIELD TO EACH TARGET COMPLEX
C    TC    - NUMBER OF TARGET COMPLEXES
C    TT    - NUMBER OF TARGET TYPES
C    TTYPE - ARRAY STORING THE NAMES OF THE TARGET TYPES
C    W1    - WEIGHTING FACTOR FOR TFIRST
C    W2    - WEIGHTING FACTOR FOR DUR
C
C  FILES:
C
C    SILOATT.DAT
C    SILOATT.ERR
C
C  LOCAL VARIABLES:
C
C    I, J, K, KK, L, LL, Q - COUNTERS
C    IERROR  - STORES THE VALUE OF IOSTAT
C    LINE    - STORES THE CONTENTS OF A LINE FROM 'SILOATT.DAT'
C    MISS    - TEMPORARILY STORES THE NAME OF A MISSILE TYPE
C    NAME    - TEMPORARILY STORES THE NAME OF A LAUNCH FIELD OR TARGET COMPLEX
C    NUM     - TEMPORARILY STORES THE NUMBER OF MISSILES OR TARGETS
C    TARG    - TEMPORARILY STORES THE NAME OF A TARGET TYPE
C    TEMP    - ARRAY THAT TEMPORARILY STORES FLIGHT TIME DATA
C
C  ERROR MESSAGES:  NONE
C
C*******************************************************************************
C
C  LOCAL VARIABLE DECLARATIONS:
      INTEGER  I, IERROR, J, K, KK, L, LF, LL, M(MAXLF,MAXMT), MT,
     &         N(MAXTC,MAXTT), NUM, Q, TC, TT

      REAL     PCT, PCTBU, T(MAXLF,MAXTC,MAXMT), TEMP(10), W1, W2

      CHARACTER*80 LINE

      CHARACTER*8 CPLX(MAXTC), FIELD(MAXLF), NAME

      CHARACTER*4 MISS, TARG, MTYPE(MAXMT), TTYPE(MAXTT)

      I = 0
      J = 0
      K = 0
      L = 0
      LF = 0
      MT = 0
      TC = 0
      TT = 0

      OPEN(UNIT=10, FILE='siloatt.dat', ACCESS='SEQUENTIAL',
     &     STATUS='OLD', IOSTAT=IERROR, ERR=9991)

   10 READ(10,1000,END=500) LINE

C IF THE FIRST CHARACTER OF A LINE IS BLANK, READ THE NEXT LINE.
      IF (LINE(1:1) .EQ. ' ') GO TO 10

C IF THE FIRST 12 CHARACTERS OF THE LINE ARE 'MISSILE DATA', THEN THE
C FOLLOWING LINES GIVE THE NUMBER OF MISSILES BY TYPE AT EACH LAUNCH FIELD.
      IF (LINE(1:12) .EQ. 'MISSILE DATA') THEN

C SKIP THE NEXT SIX LINES:
          DO 20 Q = 1, 6
              READ(10,*)
   20     CONTINUE
   25     READ(10,1010,END=500) NAME, MISS, NUM

C IF THE FIRST CHARACTER OF A LINE IS BLANK, THIS SECTION IS DONE.
          IF (NAME(1:1) .EQ. ' ') GO TO 10

C OTHERWISE, READ IN THE DATA.  IF NO LAUNCH FIELDS HAVE BEEN DETERMINED,
C MAKE THIS ONE THE FIRST ONE.
          IF (I .EQ. 0) THEN
              I = 1
              FIELD(I) = NAME
              LF = LF + 1

C ELSE IF THIS IS THE SAME LAUNCH FIELD AS BEFORE, CHECK THE MISSILE TYPE.
          ELSE IF (NAME .EQ. FIELD(I)) THEN
```

```
              GO TO 30

C ELSE, THIS IS A NEW LAUNCH FIELD, SO ADD ITS NAME TO THE LIST:
          ELSE
              I = I + 1
              FIELD(I) = NAME
              LF = LF + 1
          END IF

C CHECK THE MISSILE TYPE.  IF THIS IS THE FIRST ONE, ADD IT TO THE LIST:
   30     IF (K .EQ. 0) THEN
              K = 1
              MTYPE(K) = MISS
              MT = MT + 1

C ELSE, CHECK THE LIST.  IF THIS TYPE IS ALREADY IN THE LIST, ADD THE NUMBER
C OF MISSILES TO THE APPROPRIATE LAUNCH FIELD.
          ELSE
              DO 40 KK = 1, MT
                  IF (MISS .EQ. MTYPE(KK)) THEN
                      K = KK
                      GO TO 50
                  END IF
   40         CONTINUE
              K = MT + 1
              MTYPE(K) = MISS
              MT = MT + 1
          END IF

   50     M(I,K) = NUM
          GO TO 25

C IF THE FIRST 11 CHARACTERS OF THE LINE ARE 'TARGET DATA', THIS SECTION
C CONTAINS THE NUMBER OF TARGETS BY TYPE AT EACH TARGET COMPLEX:
      ELSE IF (LINE(1:11) .EQ. 'TARGET DATA') THEN

C SKIP THE NEXT FOUR LINES:
          DO 60 Q = 1, 4
              READ(10,*)
   60     CONTINUE
   65     READ(10,1010,END=500) NAME, TARG, NUM

C IF THE FIRST CHARACTER OF A LINE IS BLANK, THIS SECTION IS DONE.
          IF (NAME(1:1) .EQ. ' ') GO TO 10

C OTHERWISE, IF THIS IS THE FIRST TARGET COMPLEX, ADD ITS NAME TO THE LIST:
          IF (J .EQ. 0) THEN
              J = 1
              CPLX(J) = NAME
              TC = TC + 1
```

```
C ELSE IF THIS COMPLEX IS ALREADY IN THE LIST, CHECK THE TARGET TYPE.
          ELSE IF (NAME .EQ. CPLX(J)) THEN
              GO TO 70

C ELSE, THIS IS A NEW TARGET COMPLEX, SO ADD IT TO THE LIST:
          ELSE
              J = J + 1
              CPLX(J) = NAME
              TC = TC + 1
          END IF

C CHECK THE TARGET TYPE.  IF THIS IS THE FIRST ONE, ADD IT TO THE LIST.
   70     IF (L .EQ. 0) THEN
              L = 1
              TTYPE(L) = TARG
              TT = TT + 1

C OTHERWISE, CHECK THE LIST TO SEE IF THIS ONE IS ALREADY ON IT.
          ELSE
              DO 80 LL = 1, TT
                  IF (TARG .EQ. TTYPE(LL)) THEN
                      L = LL
                      GO TO 90
                  END IF
   80         CONTINUE
              L = TT + 1
              TTYPE(L) = TARG
              TT = TT + 1
          END IF

   90     N(J,L) = NUM
          GO TO 65

C IF THE FIRST 16 CHARACTERS OF THE LINE ARE 'FLIGHT TIME DATA', THE NEXT
C SECTION LISTS THE FLIGHT TIMES OF MISSILES (BY TYPE) FROM EACH LAUNCH
C FIELD TO EACH TARGET COMPLEX.
      ELSE IF (LINE(1:16) .EQ. 'FLIGHT TIME DATA') THEN
          DO 100 Q = 1, 5
              READ(10,*)
  100     CONTINUE
  110     READ(10,1020,END=500) NAME, MISS, (TEMP(J), J = 1, TC)

C IF THE FIRST CHARACTER OF A LINE IS BLANK, THIS SECTION IS DONE:
          IF (NAME(1:1) .EQ. ' ') GO TO 10

C FIND THE CURRENT NAME IN THE LIST OF LAUNCH FIELDS:
          DO 120 Q = 1, LF
              IF (NAME .EQ. FIELD(Q)) I = Q
  120     CONTINUE

C FIND THE CURRENT MISSILE TYPE IN THE LIST OF MISSILE TYPES:
        DO 130 Q = 1, MT
            IF (MISS .EQ. MTYPE(Q)) K = Q
 130    CONTINUE

C ASSIGN THE FLIGHT TIMES FOR TYPE K MISSILES FROM LAUNCH FIELD I TO EACH
C TARGET COMPLEX J:
        DO 140 J = 1, TC
            T(I,J,K) = TEMP(J)
 140    CONTINUE
        GO TO 110

C IF THE FIRST NINE CHARACTERS OF THE LINE ARE 'WEIGHTING', THEN THE NEXT
C SECTION CONTAINS THE WEIGHTING FACTORS:
      ELSE IF (LINE(1:9) .EQ. 'WEIGHTING') THEN
        READ(10,*)
        READ(10,1030) W1
        READ(10,1030) W2

C IF THE FIRST 10 CHARACTERS OF THE LINE ARE 'PERCENTAGE', THE NEXT SECTION
C SPECIFIES PCT:
      ELSE IF (LINE(1:10) .EQ. 'PERCENTAGE') THEN
        READ(10,*)
        READ(10,1040) PCT

C IF THE FIRST 14 CHARACTERS OF THE LINE ARE 'PERCENT BACKUP', THE NEXT
C SECTION SPECIFIES PCTBU:
      ELSE IF (LINE(1:14) .EQ. 'PERCENT BACKUP') THEN
        READ(10,*)
        READ(10,1050) PCTBU
        GO TO 500

C OTHERWISE, THE FILE 'SILOATT.DAT' HAS AN ERROR OR IS THE WRONG FILE:
      ELSE
        OPEN(UNIT=15, FILE='SILOATT.ERR', STATUS='UNKNOWN',
     &       ACCESS='SEQUENTIAL')
        WRITE(15,*) 'THIS IS THE WRONG INPUT FILE'
        CLOSE(15)
        STOP
      END IF

      GO TO 10

 500  CLOSE(10)

      RETURN

9991  OPEN(UNIT=15, FILE='SILOATT.ERR', STATUS='UNKNOWN',
     &     ACCESS='SEQUENTIAL')
      WRITE(15,*) 'ERROR IN OPENING FILE'
      WRITE(15,*) 'ERROR CODE = ', IERROR
      CLOSE(15)
      STOP

 1000 FORMAT(A80)
 1010 FORMAT(A8, 1X, A4, 4X, I3)
 1020 FORMAT(A8, 1X, A4, 4X, 10(F6.2, 3X))
 1030 FORMAT(5X, F6.2)
 1040 FORMAT(6X, F4.2)
 1050 FORMAT(8X, F4.2)

      END

C*******************************************************************************
C*******************************************************************************
      SUBROUTINE PROCESS(BU, CPLX, LF, LFMT, M, MADJ, MAXLF, MAXMT,
     &                   MAXTC, MAXTT, MT, N, NADJ, NUMBU, NUMFAIR,
     &                   NUMGOOD, NUMPOOR, NUMTAR, PCTBU, T, TC,
     &                   TCTT, TMAX, TMIN, TT)
C*******************************************************************************
C
C NAME:  PROCESS
C
C REV  DATE OF
C NO.  CHANGE   DESIGNER      CCR  DESCRIPTION
C ---  -------  ------------- ---- ----------------------------------------
C 000   9FEB93  R. PACE            ORIGINAL RELEASE
C
C DESCRIPTION:  PROCESSES THE DATA FOR THE SILO ATTACK PLAN
C
C CALLED BY:  SILOATT
C
C CALLS:  NONE
C
C CALLING SEQUENCE:
C
CIN  LF       - NUMBER OF LAUNCH FIELDS
C    M        - ARRAY STORING NUMBER OF MISSILES BY TYPE AT EACH LAUNCH FIELD
C    MAXLF    - MAX NUMBER OF LAUNCH FIELDS
C    MAXMT    - MAX NUMBER OF MISSILE TYPES
C    MAXTC    - MAX NUMBER OF TARGET COMPLEXES
C    MAXTT    - MAX NUMBER OF TARGET TYPES
C    MT       - THE NUMBER OF MISSILE TYPES
C    N        - ARRAY STORING NUMBER OF TARGETS BY TYPE AT EACH TARGET COMPLEX
C    PCTBU    - THE PERCENT OF THE AVAILABLE MISSILES AT EACH LAUNCH FIELD
C               THAT MUST BE RESERVED FOR BACKUP
C    TT       - NUMBER OF TARGET TYPES
CI/O CPLX     - ARRAY STORING THE NAMES OF THE TARGET COMPLEXES
C    TC       - NUMBER OF TARGET COMPLEXES
C    T        - ARRAY STORING FLIGHT TIMES OF MISSILES (BY TYPE) FROM EACH LAUNCH
C               FIELD TO EACH TARGET COMPLEX
COUT BU       - ARRAY STORING THE INDICES OF THE BACKUP LAUNCH FIELDS
C    LFMT     - THE NUMBER OF LAUNCH FIELD/MISSILE TYPE COMBINATIONS THAT ARE
```

```
C              POSSIBLE (I.E., THE NUMBER OF MISSILES OF THAT TYPE IS NOT ZERO)
C    MADJ    - ARRAY STORING THE ADJUSTED NUMBER OF MISSILES OF EACH TYPE AT
C              EACH LAUNCH FIELD
C    NADJ    - ARRAY STORING THE ADJUSTED NUMBER OF TARGETS OF EACH TYPE AT
C              EACH TARGET COMPLEX
C    NUMBU   - THE NUMBER OF BACKUP LAUNCH FIELDS
C    NUMFAIR - THE NUMBER OF TARGET COMPLEXES THAT DO NOT CONTAIN ANY GOOD
C              TARGETS BUT DO CONTAIN AT LEAST ONE FAIR TARGET
C    NUMGOOD - THE NUMBER OF TARGET COMPLEXES THAT CONTAIN AT LEAST ONE GOOD
C              TARGET
C    NUMPOOR - THE NUMBER OF TARGET COMPLEXES THAT DO NOT CONTAIN ANY GOOD OR
C              FAIR TARGETS
C    NUMTAR  - ARRAY STORING THE TOTAL NUMBER OF TARGETS AT EACH TARGET COMPLEX
C    TCTT    - THE NUMBER OF TARGET COMPLEX/TARGET TYPE COMBINATIONS THAT ARE
C              POSSIBLE
C    TMAX    - THE LARGEST POSSIBLE FLIGHT TIME
C    TMIN    - THE SMALLEST POSSIBLE FLIGHT TIME
C
C  FILES:
C
C    SILOATT.DAT
C    SILOATT.ERR
C
C  LOCAL VARIABLES:
C
C    ANS1    - STORES RESPONSE TO WHETHER OR NOT ANY LAUNCH FIELDS ARE FOR
C              BACKUP ONLY
C    ANS2    - STORES RESPONSE TO WHETHER OR NOT ANY TARCET COMPLEXES SHOULD BE
C              COMBINED
C    COMB    - ARRAY STORING ID OF TARGET COMPLEXES TO BE COMBINED
C    LINE    - STORES THE CONTENTS OF A LINE FROM 'SILOATT.DAT'
C    NUMTGT1 - THE TOTAL NUMBER OF TARGETS AT THE FIRST COMPLEX TO BE COMBINED
C    NUMTGT2 - THE TOTAL NUMBER OF TARGETS AT THE SECOND COMPLEX TO BE COMBINED
C    TEMP    - ARRAY USED TO TEMPORARILY STORE THE RESULT OF MULTIPLYING EACH
C              N(J,L) BY 0.1
C
C  ERROR MESSAGES:  NONE
C
C******************************************************************************
C
C  LOCAL VARIABLE DECLARATIONS:

      INTEGER  BU(MAXLF), COMB(2), LF, LFMT, M(MAXLF,MAXMT),
     &         MADJ(MAXLF,MAXMT), MT, N(MAXTC,MAXTT), NADJ(MAXTC,MAXTT),
     &         NUMBU, NUMFAIR, NUMGOOD, NUMPOOR, NUMTAR(MAXTC), NUMTGT1,
     &         NUMTGT2, TC, TCTT, TT

      REAL  PCTBU, T(MAXLF,MAXTC,MAXMT), TEMP(10,5), TMAX, TMIN

      CHARACTER  ANS1, ANS2
      CHARACTER*8 CPLX(MAXTC)

      CHARACTER*80 LINE

      OPEN(UNIT=10, FILE='SILOATT.DAT', STATUS='OLD',
     &    ACCESS='SEQUENTIAL', IOSTAT=IERROR, ERR=991)

   10 READ(10,1020) LINE

C IF THE FIRST THREE CHARACTERS OF A LINE ARE 'ARE', THE NEXT SECTION TELLS
C WHETHER OR NOT ANY LAUNCH FIELDS ARE FOR BACKUP ONLY.
      IF (LINE(1:3) .EQ. 'ARE') THEN
          READ(10,1000) ANS1

C IF ANY LAUNCH FIELDS ARE FOR BACKUP ONLY, READ IN HOW MANY AND WHICH ONES:
          IF ((ANS1 .EQ. 'Y') .OR. (ANS1 .EQ. 'y')) THEN
              READ(10,*)
              READ(10,*)
              READ(10,*) NUMBU
              READ(10,*)
              READ(10,*)
              READ(10,*) (BU(I), I = 1, NUMBU)

C OTHERWISE, SKIP THE NEXT SIX LINES:
          ELSE
              DO 20 I = 1, 6
                  READ(10,*)
   20         CONTINUE
          END IF

          READ(10,*)
          READ(10,*)
          READ(10,1000) ANS2

C IF ANY OF THE TARGET COMPLEXES SHOULD BE COMBINED, READ IN WHICH ONES AND
C WHAT THE NAME OF THE COMBINED COMPLEX SHOULD BE:
          IF ((ANS2 .EQ. 'Y') .OR. (ANS2 .EQ. 'y')) THEN
              READ(10,*)
              READ(10,*)
              READ(10,*) (COMB(I), I=1, 2)
              READ(10,*)
              READ(10,*)
              READ(10,1010) CPLX(COMB(1))

              NUMTGT1 = 0
              NUMTGT2 = 0

C THIS DO LOOP CALCULATES THE TOTAL NUMBER OF TARGETS AT THE TWO TARGET
C COMPLEXES THAT WILL BE COMBINED:
              DO 30 L = 1, TT
                  NUMTGT1 = NUMTGT1 + N(COMB(1),L)
                  NUMTGT2 = NUMTGT2 + N(COMB(2),L)
   30          CONTINUE

C THIS IMBEDDED DO LOOP CALCULATES THE WEIGHTED AVERAGE OF THE FLIGHT TIMES
C FOR EACH LAUNCH FIELD/MISSILE TYPE COMBINATION TO THE TWO TARGET COMPLEXES
C THAT ARE BEING COMBINED:
               DO 50 I = 1, LF
                  DO 40 K = 1, MT
                     T(I,COMB(1),K) = (NUMTGT1*T(I,COMB(1),K)
     &                 + NUMTGT2*T(I,COMB(2),K))/(NUMTGT1 + NUMTGT2)
   40             CONTINUE
   50          CONTINUE

C HERE THE PROGRAM WILL TREAT THE COMBINED TARGET COMPLEX AS HAVING THE LOWER
C INDEX OF THE TWO TARGET COMPLEXES IT REPRESENTS, AND "SLIDES" THE OTHERS
C FORWARD AS APPROPRIATE:

C IF THE LARGER OF THE TWO INDICES IS NOT THE LAST TARGET COMPLEX, THEN
C SLIDE THE FLIGHT TIME DATA FORWARD FOR EACH TARGET COMPLEX WITH A BIGGER
C INDEX:
               IF (COMB(2) .NE. TC) THEN
                  DO 80 I = 1, LF
                     DO 70 J = COMB(2), TC-1
                        DO 60 K = 1, MT
                           T(I,J,K) = T(I,J+1,K)
   60                   CONTINUE
                        CPLX(J) = CPLX(J+1)
   70                CONTINUE
   80             CONTINUE
               END IF

C THIS DO LOOP STORES THE NUMBER OF TARGETS OF EACH TYPE OF THE COMBINED
C COMPLEX INTO THE ARRAY PREVIOUSLY USED FOR THE COMPLEX WITH THE LOWER INDEX:
               DO 90 L = 1, TT
                  N(COMB(1),L) = N(COMB(1),L) + N(COMB(2),L)
   90          CONTINUE

C IF THE COMPLEX WITH THE LARGER INDEX IS NOT THE LAST TARGET COMPLEX, THEN
C THE DO LOOP SLIDES THE TARGET DATA FORWARD FOR ALL COMPLEXES WITH A LARGER
C INDEX:
               IF (COMB(2) .NE. TC) THEN
                  DO 110 J = COMB(2), TC-1
                     DO 100 L = 1, TT
                        N(J,L) = N(J+1,L)
  100                CONTINUE
  110             CONTINUE
               END IF
               TC = TC - 1
            END IF
            GO TO 120
```

```
      ELSE
          GO TO 10
      END IF

C HERE THE PROGRAM CALCULATES THE NUMBER OF TARGET FIELDS THAT HAVE
C GOOD, FAIR, AND POOR TARGETS RESPECTIVELY:
  120 NUMGOOD = 0
      NUMFAIR = 0
      NUMPOOR = 0

C THIS DO LOOP ITERATES THROUGH EACH TARGET COMPLEX:
      DO 130 J = 1, TC

C IF THERE ARE GOOD TARGETS AT COMPLEX J, THEN INCREASE NUMGOOD BY 1:
          IF (N(J,1) .GT. 0) THEN
              NUMGOOD = NUMGOOD + 1

C ELSE, IF THERE ARE FAIR TARGETS AT COMPLEX J, THEN INCREASE NUMFAIR BY 1:
          ELSEIF (N(J,2) .GT. 0) THEN
              NUMFAIR = NUMFAIR + 1

C ELSE, THERE ARE ONLY POOR TARGETS AT COMPLEX J, SO INCREASE NUMPOOR BY 1:
          ELSE
              NUMPOOR = NUMPOOR + 1
          END IF
  130 CONTINUE

C HERE THE PROGRAM FINDS TMIN (THE SMALLEST FLIGHT TIME) AND TMAX
C (THE LARGEST FLIGHT TIME):
      TMIN = 1000.0
      TMAX = 0.0

C THIS IMBEDDED DO LOOP STEPS THROUGH ALL COMBINATIONS OF LAUNCH FIELD/
C TARGET COMPLEX/MISSILE TYPE LOOKING FOR THE LARGEST AND SMALLEST T(I,J,K):
      DO 160 I = 1, LF
          DO 150 J = 1, TC
              DO 140 K = 1, MT

C IF THERE ARE NO MISSILES OF TYPE K AT LAUNCH FIELD I, T(I,J,K) WILL BE 0,
C TRY THE NEXT MISSILE TYPE:
                  IF (T(I,J,K) .EQ. 0.0) GO TO 140

C IF THE CURRENT T(I,J,K) IS GREATER THAN THE CURRENT TMAX, SET TMAX EQUAL TO
C THE CURRENT T(I,J,K):
                  IF (T(I,J,K) .GT. TMAX) TMAX = T(I,J,K)

C IF THE CURRENT T(I,J,K) IS LESS THAN THE CURRENT TMIN, SET TMIN EQUAL TO
C THE CURRENT T(I,J,K):
                  IF (T(I,J,K) .LT. TMIN) TMIN = T(I,J,K)
  140         CONTINUE
  150     CONTINUE
  160 CONTINUE

C HERE THE PROGRAM CALCULATES THE NUMBER OF LAUNCH FIELD/MISSILE
C TYPE COMBINATIONS EXIST:
      LFMT = 0

C STEP THROUGH EACH LAUNCH FIELD:
      DO 190 I = 1, LF

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 170 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD (I) IS ONE OF THE BACKUP ONES, GO TO THE
C NEXT LAUNCH FIELD:
              IF (I .EQ. BU(P)) GO TO 190
  170     CONTINUE

C STEP THROUGH EACH MISSILE TYPE:
          DO 180 K = 1, MT

C IF THERE ARE NO MISSILES OF TYPE K AT LAUNCH FIELD I, GO TO THE NEXT
C MISSILE TYPE.  OTHERWISE, INCREASE LFMT BY ONE:
              IF (M(I,K) .EQ. 0) THEN
                  GO TO 180
              ELSE
                  LFMT = LFMT + 1
              END IF
  180     CONTINUE
  190 CONTINUE
      TCTT = 0

C STEP THROUGH EACH TARGET COMPLEX:
      DO 210 J = 1, TC

C STEP THROUGH EACH TARGET TYPE:
          DO 200 L = 1, TT

C IF THERE ARE NO TARGETS OF TYPE L AT TARGET COMPLEX J, GO TO THE NEXT
C TARGET TYPE.  OTHERWISE, INCREASE TCTT BY 1:
              IF (N(J,L) .EQ. 0) THEN
                  GO TO 200
              ELSE
                  TCTT = TCTT + 1
              END IF
  200     CONTINUE
  210 CONTINUE

C HERE THE PROGRAM ADJUSTS THE NUMBER OF MISSILES AT EACH LAUNCH FIELD.

C FOR FAIR MISSILES, THE PROGRAM WILL ONLY CONSIDER 50% (ROUNDED DOWN)
C OF THE MISSILES AVAILABLE FOR THE ALLOCATION:
```

```
C STEP THROUGH EACH LAUNCH FIELD:
      DO 230 I = 1, LF

C STEP THROUGH EACH MISSILE TYPE:
          DO 220 K = 1, MT

C SET THE MADJ(I,K) = M(I,K):
              MADJ(I,K) = M(I,K)
  220     CONTINUE
  230 CONTINUE

C******************************************************************************
C IF YOU DO NOT WISH TO ENFORCE THAT EACH TARGET ATTACKED BY A FAIR TARGET IN
C THE INITIAL WAVE BE ATTACKED BY A FAIR MISSILE FROM THE SAME LAUNCH FIELD IN
C THE SECOND WAVE, THEN COMMENT OUT THIS SECTION:

C STEP THROUGH EACH LAUNCH FIELD:
      DO 240 I = 1, LF

C FOR FAIR MISSILES (K = 2), SET MADJ(I,2) EQUAL TO ONE-HALF OF M(I,2),
C ROUNDING DOWN FOR FRACTIONS:
          MADJ(I,2) = INT(0.5*MADJ(I,2))
  240 CONTINUE

C******************************************************************************

C******************************************************************************
C IF YOU DO NOT WISH TO HAVE A PCTBU BACKUP OF EACH MISSILE TYPE AT EACH
C LAUNCH FIELD, THEN COMMENT OUT THIS SECTION:

C SINCE THERE SHOULD BE A PCTBU BACKUP OF EACH MISSILE TYPE AT EACH LAUNCH
C FIELD, MULTIPLY BY (1 - PCTBU) AND ROUND DOWN:

C STEP THROUGH EACH LAUNCH FIELD:
      DO 260 I = 1, LF

C STEP THROUGH EACH MISSILE TYPE:
          DO 250 K = 1, MT

C SET MADJ(I,K) EQUAL TO 0.9 TIMES MADJ(I,K), ROUNDING DOWN FOR FRACTIONS:
              MADJ(I,K) = INT((1.0 - PCTBU)*MADJ(I,K))
  250     CONTINUE
  260 CONTINUE

C******************************************************************************

C HERE THE PROGRAM DIVIDES THE NUMBER OF TARGETS AT EACH COMPLEX BY 10,
C ROUNDING UP FOR ANY FRACTION:

C******************************************************************************
```

```
C IF EACH TYPE OF MISSILE CAN HAVE A DIFFERENT NUMBER OF WARHEADS, THEN COMMENT
C OUT THIS SECTION AND INCLUDE THE NEXT ONE:

C STEP THROUGH EACH TARGET COMPLEX:
      DO 280 J = 1, TC

C STEP THROUGH EACH TARGET TYPE:
          DO 270 L = 1, TT
              TEMP(J,L) = 0.1*N(J,L)

C IF 0.1*N(J,L) IS NOT AN INTEGER, ROUND UP FOR THE FRACTION:
              IF (AMOD(TEMP(J,L),1.0) .GT. 0) THEN
                  NADJ(J,L) = INT(TEMP(J,L)) + 1

C OTHERWISE, SET NADJ(J,L) = 0.1*N(J,L):
              ELSE
                  NADJ(J,L) = INT(TEMP(J,L))
              END IF
  270     CONTINUE
  280 CONTINUE

C*******************************************************************************
C      DO 280 J = 1, TC
C          DO 270 L = 1, TT
C              NADJ(J,L) = N(J,L)
C  270     CONTINUE
C  280 CONTINUE
C*******************************************************************************

C CALCULATE THE NUMBER OF TARGETS AT EACH COMPLEX:

C STEP THROUGH EACH TARGET COMPLEX:
      DO 300 J = 1, TC
          NUMTAR(J) = 0

C STEP THROUGH EACH TARGET TYPE, SETTING THE NUMBER OF TARGETS AT COMPLEX J
C EQUAL TO THE CURRENT NUMBER PLUS THE ADJUSTED NUMBER OF TARGETS OF TYPE L
C AT COMPLEX J:
          DO 290 L = 1, TT
              NUMTAR(J) = NUMTAR(J) + NADJ(J,L)
  290     CONTINUE
  300 CONTINUE

      CLOSE(10)
      RETURN

  991 OPEN(UNIT=15, FILE='SILOATT.ERR', ACCESS='SEQUENTIAL',
     &     STATUS='UNKNOWN')
      WRITE(15,*) 'ERROR IN OPENING FILE SILOATT.DAT'
      WRITE(15,*) 'ERROR CODE = ', IERROR
      CLOSE(15)
      STOP

 1000 FORMAT(A1)
 1010 FORMAT(A8)
 1020 FORMAT(A80)

      END

C*****************************************************************************
C*****************************************************************************
      SUBROUTINE FORMUL(BU, LF, LFMT, MADJ, MAXLF, MAXMT, MAXTC, MAXTT,
     &                  MT, NADJ, NUMBU, NUMFAIR, NUMGOOD, NUMPOOR,
     &                  NUMTAR, PCT, T, TC, TCTT, TMAX, TMIN, TT,
     &                  W1, W2)
C*****************************************************************************
C
C  NAME:  FORMUL
C
C  REV  DATE OF
C  NO.  CHANGE   DESIGNER     CCR   DESCRIPTION
C  ---  -------  ------------ ----  ----------------------------------------
C  000   9FEB93  R. PACE            ORIGINAL RELEASE
C
C  DESCRIPTION:  WRITES THE DATA INTO AN MPS-FORMAT FILE
C
C  CALLED BY:  SILOATT
C
C  CALLS:  NONE
C
C  CALLING SEQUENCE:
C
CIN  BU      - ARRAY STORING THE INDICES OF THE BACKUP LAUNCH FIELDS
C    LF      - NUMBER OF LAUNCH FIELDS
C    LFMT    - THE NUMBER OF LAUNCH FIELD/MISSILE TYPE COMBINATIONS THAT ARE
C              POSSIBLE (I.E., THE NUMBER OF MISSILES OF THAT TYPE IS NOT ZERO)
C    MADJ    - ARRAY STORING THE ADJUSTED NUMBER OF MISSILES OF EACH TYPE AT
C              EACH LAUNCH FIELD
C    MAXLF   - MAX NUMBER OF LAUNCH FIELDS
C    MAXMT   - MAX NUMBER OF MISSILE TYPES
C    MAXTC   - MAX NUMBER OF TARGET COMPLEXES
C    MAXTT   - MAX NUMBER OF TARGET TYPES
C    MT      - THE NUMBER OF MISSILE TYPES
C    NADJ    - ARRAY STORING THE ADJUSTED NUMBER OF TARGETS OF EACH TYPE AT
C              EACH TARGET COMPLEX
C    NUMBU   - THE NUMBER OF BACKUP LAUNCH FIELDS
C    NUMFAIR - THE NUMBER OF TARGET COMPLEXES THAT DO NOT CONTAIN ANY GOOD
C              TARGETS BUT DO CONTAIN AT LEAST ONE FAIR TARGET
C    NUMGOOD - THE NUMBER OF TARGET COMPLEXES THAT CONTAIN AT LEAST ONE GOOD
C              TARGET
C    NUMPOOR - THE NUMBER OF TARGET COMPLEXES THAT DO NOT CONTAIN ANY GOOD OR
C              FAIR TARGETS
```

```
C    NUMTAR  - ARRAY STORING THE TOTAL NUMBER OF TARGETS AT EACH TARGET COMPLEX
C    PCT     - PERCENT OF THE MINIMUM FLIGHT TIME THAT THE DURATION IS ALLOWED
C              TO BE
C    T       - ARRAY STORING FLIGHT TIMES OF MISSILES (BY TYPE) FROM EACH LAUNCH
C              FIELD TO EACH TARGET COMPLEX
C    TC      - NUMBER OF TARGET COMPLEXES
C    TCTT    - THE NUMBER OF TARGET COMPLEX/TARGET TYPE COMBINATIONS THAT ARE
C              POSSIBLE
C    TMAX    - THE LARGEST POSSIBLE FLIGHT TIME
C    TMIN    - THE SMALLEST POSSIBLE FLIGHT TIME
C    TT      - NUMBER OF TARGET TYPES
C    W1      - THE RELATIVE WEIGHT OF MINIMIZING THE EARLIEST FLIGHT TIME
C    W2      - THE RELATIVE WEIGHT OF MINIMIZING THE DURATION OF THE ATTACK
C
C  FILES:  NONE
C
C  LOCAL VARIABLES:
C
C    COUNT      - COUNTER
C    COUNT1     - COUNTER
C    I, J, K, L - COUNTERS
C    IERROR     - STORES THE VALUE OF IOSTAT
C    ITERLIM    - UPPER LIMIT ON THE NUMBER OF LP ITERATIONS TO BE PERFORMED
C                 BY ZOOM
C    MAXNODES   - UPPER LIMIT ON THE NUMBER OF NODES ALLOWED IN THE BRANCH AND
C                 BOUND TREE IN ZOOM
C    MAXSAVE    - UPPER LIMIT ON THE NUMBER OF WARM BASES SAVED DURING THE
C                 BRANCH BOUND SEARCH IN ZOOM
C    NUMINT     - NUMBER OF 0/1 VARIABLES IN THE MIP
C    NUMROWS    - NUMBER OF ROWS IN THE MIP
C    NUMVARS    - NUMBER OF VARIABLES IN THE MIP (EXCLUDING SLACKS)
C    P, Q       - COUNTERS
C    R - R7     - COUNTERS
C
C  ERROR MESSAGES:  NONE
C
C******************************************************************************
C
C  LOCAL VARIABLE DECLARATIONS:

      INTEGER  BU(MAXLF), COUNT, COUNT1, I, IERROR, ITERLIM, J, K, L, LF,
     &         LFMT, MADJ(MAXLF,MAXMT), MAXNODES, MAXSAVE, MT,
     &         NADJ(MAXTC,MAXTT),NUMBU, NUMFAIR, NUMGOOD, NUMINT,
     &         NUMPOOR, NUMROWS, NUMTAR(MAXTC),  NUMVARS, P, Q, R, R1,
     &         R2, R3, R4, R5, R6, R7, TC, TCTT, TT

      REAL     PCT, T(MAXLF,MAXTC,MAXMT), TMAX, TMIN, W1, W2


      OPEN(UNIT=40, FILE='SILOATT.MPS', ACCESS='SEQUENTIAL',
     &     STATUS='UNKNOWN', IOSTAT=IERROR, ERR=991)

C THE FIRST LINE CONTAINS THE NAME OF THE MODEL.  THIS IS
C FOLLOWED BY THE 'ROWS' SECTION:
   10 WRITE(40,1000), 'NAME', 'SILO ATTACK PLAN( MIN)', 'ROWS',
     &                   'N', 'OBJ'
      Q = 0

C STEP THROUGH EACH TARGET COMPLEX:
      DO 30 J = 1, TC

C STEP THROUGH EACH TARGET TYPE:
         DO 20 L = 1, TT

C IF THERE ARE TARGETS OF TYPE L AT TARGET COMPLEX J, THEN WRITE A "ROWS" LINE
C FOR AN EQUALITY CONSTRAINT FOR THAT TARGET COMPLEX/TARGET TYPE COMBINATION:
            IF (NADJ(J,L) .GT. O) THEN
               Q = Q + 1
               WRITE(40,1010), 'E', 'C', Q
            END IF
   20    CONTINUE
   30 CONTINUE

C STEP THROUGH EACH LAUNCH FIELD:
      DO 60 I = 1, LF

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
         DO 40 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS A BACKUP ONE, GO TO THE NEXT FIELD:
            IF (I .EQ. BU(P)) GO TO 60
   40    CONTINUE

C STEP THROUGH EACH MISSILE TYPE:
         DO 50 K = 1, MT

C IF THERE ARE MISSILES OF TYPE K AT LAUNCH FIELD I, THEN WRITE A "ROWS" LINE
C FOR A LESS THAN OR EQUAL TO CONSTRAINT FOR THAT LAUNCH FIELD/MISSILE TYPE
C COMBINATION:
            IF (MADJ(I,K) .GT. O) THEN
               Q = Q + 1
               WRITE(40,1010), 'L', 'C', Q
            END IF
   50    CONTINUE
   60 CONTINUE

C STEP THROUGH EACH LAUNCH FIELD:
      DO 100 I = 1, LF

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
         DO 70 P = 1, NUMBU
```

```
C IF THE CURRENT LAUNCH FIELD IS A BACKUP LAUNCH FIELD, GO THE NEXT ONE:
              IF (I .EQ. BU(P)) GO TO 100
   70     CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
          DO 90 J = 1, TC

C STEP THROUGH EACH MISSILE TYPE:
              DO 80 K = 1, MT

C IF THERE ARE MISSILES OF TYPE K AT LAUNCH FIELD I, THEN WRITE A "ROWS" LINE
C FOR A LESS THAN OR EQUAL TO CONSTRAINT FOR THAT LAUNCH FIELD/TARGET COMPLEX/
C MISSILE TYPE COMBINATION:

                  IF (MADJ(I,K) .GT. 0) THEN
                      Q = Q + 1
                      WRITE(40,1010), 'L', 'C', Q
                  END IF
   80         CONTINUE
   90     CONTINUE
  100 CONTINUE

C STEP THROUGH EACH LAUNCH FIELD:
      DO 140 I = 1, LF

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 110 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS A BACKUP LAUNCH FIELD, GO TO THE NEXT ONE:
              IF (I .EQ. BU(P)) GO TO 140
  110     CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
          DO 130 J = 1, TC

C STEP THROUGH EACH MISSILE TYPE:
              DO 120 K = 1, MT

C IF THERE ARE MISSILES OF TYPE K AT LAUNCH FIELD I, THEN WRITE A "ROWS" LINE
C FOR A GREATER THAN OR EQUAL TO CONSTRAINT FOR THAT LAUNCH FIELD/TARGET
C COMPLEX/MISSILE TYPE COMBINATION:
                  IF (MADJ(I,K) .GT. 0) THEN
                      Q = Q + 1
                      WRITE(40,1010), 'G', 'C', Q
                  END IF
  120         CONTINUE
  130     CONTINUE
  140 CONTINUE

C******************************************************************************
```

```
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, COMMENT OUT THIS
C SECTION AND INCLUDE THE NEXT:

C STEP THROUGH EACH LAUNCH FIELD/TARGET COMPLEX COMBINATION (EXCLUDING
C BACKUPS), AND WRITE A "ROWS" LINE FOR A LESS THAN OR EQUAL TO CONSTRAINT
C FOR EACH ONE:
      DO 150 P = 1, (LF - NUMBU)*TC
          Q = Q + 1
          WRITE(40,1010), 'L', 'C', Q
  150 CONTINUE

C*******************************************************************************
C      DO 155 I = 1, LF
C          DO 150 P = 1, NUMBU
C              IF (I .EQ. BU(P)) GO TO 155
C  150     CONTINUE
C          DO 153 J = 1, TC
C              DO 152 K = 1, MT
C                  IF (MADJ(I,K) .EQ. 0) GO TO 152
C                  Q = Q + 1
C                  WRITE(40,1010), 'L', 'C', Q
C  152         CONTINUE
C  153     CONTINUE
C*******************************************************************************

C*******************************************************************************
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, COMMENT OUT THIS
C SECTION:

C STEP THROUGH EACH LAUNCH FIELD/TARGET COMPLEX COMBINATION (EXCLUDING
C BACKUPS), AND WRITE A "ROWS" LINE FOR A GREATER THAN OR EQUAL TO CONSTRAINT
C FOR EACH ONE:
      DO 160 P = 1, (LF - NUMBU)*TC
          Q = Q + 1
          WRITE(40,1010), 'G', 'C', Q
  160 CONTINUE

C*******************************************************************************

C*******************************************************************************
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, COMMENT OUT THIS
C SECTION:

C STEP THROUGH EACH TARGET COMPLEX, AND WRITE A "ROWS" LINE FOR AN EQUALITY
C CONSTRAINT FOR EACH ONE:
      DO 170 J = 1, TC
          Q = Q + 1
C*******************************************************************************
C IF THE USER WANTS TO ALLOW A TARGET COMPLEX TO BE ATTACKED BY TWO LAUNCH
C FIELDS, COMMENT OUT THIS STATEMENT AND INCLUDE THE NEXT ONE:
          WRITE(40,1010), 'E', 'C', Q
```

```
C          WRITE(40,1010) 'L', 'C', Q
C*****************************************************************************
  170 CONTINUE

C STEP THROUGH EACH LAUNCH FIELD:
      DO 210 I = 1, LF

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 180 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS A BACKUP ONE, GO TO THE NEXT LAUNCH FIELD:
              IF (I .EQ. BU(P)) GO TO 210
  180     CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
          DO 200 J = 1, TC

C STEP THROUGH EACH MISSILE TYPE:
              DO 190 K = 1, MT

C IF THERE ARE MISSILES OF TYPE K AT LAUNCH FIELD I, THEN WRITE A "ROWS" LINE
C FOR A GREATER THAN OR EQUAL TO CONSTRAINT FOR EACH LAUNCH FIELD/TARGET
C COMPLEX/MISSILE TYPE COMBINATION:
                  IF (MADJ(I,K) .GT. 0) THEN
                      Q = Q + 1
                      WRITE(40,1010), 'G', 'C', Q
                  END IF
  190         CONTINUE
  200     CONTINUE
  210 CONTINUE

C STEP THROUGH EACH TARGET COMPLEX AND WRITE A "ROWS" LINE FOR A GREATER
C THAN OR EQUAL TO CONSTRAINT FOR EACH ONE:
      DO 220 J = 1, TC
          Q = Q + 1
          WRITE(40,1010), 'G', 'C', Q
  220 CONTINUE
      Q = Q + 1
      WRITE(40,1010), 'E', 'C', Q

C STEP THROUGH EACH TARGET COMPLEX AND WRITE A "ROWS" LINE FOR A LESS THAN
C OR EQUAL TO CONSTRAINT FOR EACH ONE:
      DO 225 J = 1, TC
          Q = Q + 1
          WRITE(40,1010), 'L', 'C', Q
  225 CONTINUE

C STEP THROUGH EACH TARGET COMPLEX AND WRITE A "ROWS" LINE FOR A GREATER THAN
C OR EQUAL TO CONSTRAINT FOR EACH ONE:
      DO 230 J = 1, TC
          Q = Q + 1
          WRITE(40,1010), 'G', 'C', Q
  230 CONTINUE
      Q = Q + 1
      WRITE(40,1010), 'E', 'C', Q
      Q = Q + 1
      WRITE(40,1010), 'L', 'C', Q

C STEP THROUGH EACH COMBINATION OF TARGET COMPLEXES WITH GOOD TARGETS AND
C TARGET COMPLEXES WITH FAIR TARGETS AND WRITE A "ROWS" LINE FOR A LESS THAN OR
C EQUAL TO CONSTRAINT FOR EACH ONE:
      DO 240 P = 1, NUMGOOD*NUMFAIR
          Q = Q + 1
          WRITE(40,1010), 'L', 'C', Q
  240 CONTINUE

C STEP THROUGH EACH COMBINATION OF TARGET COMPLEXES WITH FAIR TARGETS AND
C TARGET COMPLEXES WITH POOR TARGETS AND WRITE A "ROWS" LINE FOR A LESS THAN OR
C EQUAL TO CONSTRAINT FOR EACH ONE:
      DO 250 P = 1, NUMFAIR*NUMPOOR
          Q = Q + 1
          WRITE(40,1010), 'L', 'C', Q
  250 CONTINUE
      WRITE(40,1020), 'E', 'C', Q + 1, 'E', 'C', Q + 2

C*******************************************************************************
C IF THE USER WANTS TO ALLOW A TARGET COMPLEX TO BE ATTACKED BY TWO LAUNCH
C FIELDS, ADD IN THE FOLLOWING STATEMENT:
C      WRITE(40,1010) 'L', 'C', Q + 3
C*******************************************************************************

C*******************************************************************************
C IF THE SOLUTION IS UNBALANCED, THE USER MAY INCLUDE THE NEXT SECTION:
C      Q = Q + 1
C      DO 255 I = 1, LF
C          DO 252 P = 1, NUMBU
C              IF (I .EQ. BU(P)) GO TO 255
C              Q = Q + 1
C              WRITE(40,1010) 'L', 'C', Q
C  252     CONTINUE
C  255 CONTINUE
C*******************************************************************************

C*******************************************************************************
C IF THE USER DOES NOT WANT EACH LAUNCH FIELD TO HAVE A FCTBU BACKUP, THEN ADD
C THE NEXT SECTION (ALSO DECLARE TOTMIS):
C      TOTMIS = 0
C      DO 257 I = 1, LF
C          DO 256 K = 1, MT
C              TOTMIS = TOTMIS + MADJ(I,K)
C  256     CONTINUE
C  257 CONTINUE
```

```
C      Q = Q + 1
C      WRITE(40,1010) 'L', 'C', Q
C********************************************************************************


C THE NEXT SECTION IS THE 'COLUMNS' SECTION:
      WRITE(40,1110), 'COLUMNS'

C VARIABLE X(I,J,K,L):
      COUNT = 0
      R = TCTT
      R1 = LFMT + TCTT

C********************************************************************************
C IF THE SOLUTION IS UNBALANCED, ADD THE FOLLOWING ASSIGNMENT STATEMENT:
C      R2 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 3*TC
C     &      + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 5
C********************************************************************************

C********************************************************************************
C IF THE USER DOES NOT WANT EACH LAUNCH FIELD TO HAVE A PCTBU BACKUP, ADD THE
C FOLLOWING ASSIGNMENT STATEMENT:
C      R2 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 3*TC
C     &      + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 5
C********************************************************************************

C STEP THROUGH EACH LAUNCH FIELD:
      DO 300 I = 1, LF
          COUNT1 = 0
          Q = 0

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 260 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS A BACKUP ONE, GO TO THE NEXT LAUNCH FIELD:
              IF (I .EQ. BU(P)) GO TO 300
  260     CONTINUE
          R = R + 1

C********************************************************************************
C IF THE SOLUTION IS UNBALANCED, ADD THE FOLLOWING STATEMENT:
C          R2 = R2 + 1
C********************************************************************************

C STEP THROUGH EACH TARGET COMPLEX:
          DO 290 J = 1, TC
              Q = Q + 1
              COUNT = 0
              IF (COUNT1 .GT. 1) R = R - 1
              COUNT1 = 0
```

```
C STEP THROUGH EACH MISSILE TYPE:
              DO 280 K = 1, MT

C IF THERE ARE NO MISSILES OF TYPE K AT LAUNCH FIELD I, GO TO THE NEXT MISSILE
C TYPE:
                  IF (MADJ(I,K) .EQ. 0) GO TO 280
                  IF (COUNT .GT. 1) Q = Q - 1
                  COUNT = 0
                  COUNT1 = COUNT1 + 1
                  IF (COUNT1 .GT. 1) R = R + 1
                  R1 = R1 + 1

C STEP THROUGH EACH TARGET TYPE:
                  DO 270 L = 1, TT

C IF THERE ARE NO TARGETS OF TYPE L AT COMPLEX J, GO TO THE NEXT TARGET TYPE:
                      IF (NADJ(J,L) .EQ. 0) GO TO 270
                      COUNT = COUNT + 1
                      IF (COUNT .GT. 1) Q = Q + 1

C OTHERWISE, X(I,J,K,L) WILL BE IN CONSTRAINT Q WITH A COEFFICIENT OF 1.0, AND
C WILL BE IN CONSTRAINTS R, R1, AND R1 + LFMT*TC WITH COEFFICIENTS OF 1.0,
C -1.0, AND -1.0 RESPECTIVELY:
                      WRITE(40,1030), 'X', I, J, K, L, 'C', Q, 1.0
                      WRITE(40,1030), 'X', I, J, K, L, 'C', R, 1.0
                      WRITE(40,1030), 'X', I, J, K, L, 'C', R1, -1.0
                      WRITE(40,1030), 'X', I, J, K, L, 'C', R1 +
     &                                  LFMT*TC, -1.0

C*******************************************************************************
C IF THE SOLUTION IS UNBALANCED, DECLARE AND ASSIGN A VALUE TO JSTAR AND ADD
C THE FOLLOWING STATEMENT:
C                      IF (J .EQ. JSTAR) WRITE(40,1030) 'X', I, JSTAR,
C     &                                                K, L, 'C', R2, 1.0
C*******************************************************************************

C*******************************************************************************
C IF THE USER DOES NOT WANT TO HAVE A PCTBU AT EACH LAUNCH FIELD, ADD THE
C FOLLOWING STATEMENT:
C                      WRITE(40,1030) 'X', I, J, K, L, 'C', R2 + 1, 1.0
C*******************************************************************************

  270             CONTINUE
  280         CONTINUE
  290     CONTINUE
  300 CONTINUE
      WRITE(40,1040), 'INT'

C*******************************************************************************
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, THEN COMMENT OUT THIS
C SECTION AND INCLUDE THE NEXT.  YOU MUST ALSO DECLARE THE VARIABLES 'M' AND
```

```
C ROW(I,J,K).  YOU MUST ALSO CHANGE SOME OTHER SECTIONS AS ANNOTATED BELOW:

C VARIABLE Y(I,J,K):
      R = LFMT + TCTT
      R1 = LFMT + TCTT + 2*LFMT*TC
      R2 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + TC

C STEP THROUGH EACH LAUNCH FIELD:
      DO 340 I = 1, LF

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 310 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS A BACKUP ONE, GO TO THE NEXT LAUNCH FIELD:
              IF (I .EQ. BU(P)) GO TO 340
  310     CONTINUE
          R2 = R2 - LFMT*TC
          R3 = 0

C STEP THROUGH EACH TARGET COMPLEX:
          DO 330 J = 1, TC
              R1 = R1 + 1
              R2 = R2 - R3
              R3 = 0

C STEP THROUGH EACH MISSILE TYPE:
              DO 320 K = 1, MT

C IF THERE ARE NO MISSILES OF TYPE K AT LAUNCH FIELD I, GO TO THE NEXT MISSILE
C TYPE:
                  IF (MADJ(I,K) .EQ. 0) GO TO 320
                  R = R + 1
                  R2 = R2 + 1
                  R3 = R3 + 1

C OTHERWISE, Y(I,J,K) WILL BE IN CONSTRAINT R WITH A COEFFICIENT OF 1.0,
C CONSTRAINT R + LFMT*TC WITH A COEFFICIENT OF THE NUMBER OF TARGETS AT COMPLEX
C J, CONSTRAINT R1 WITH A COEFFICIENT OF -1.0, AND CONSTRAINT R1 +
C (LF - NUMBU)*TC WITH A COEFFICIENT OF -1.0:
                  WRITE(40,1050), 'Y', I, J, K, 'C', R, 1.0
                  WRITE(40,1050), 'Y', I, J, K, 'C', R + LFMT*TC,
     &                                 REAL(NUMTAR(J))
                  WRITE(40,1050), 'Y', I, J, K, 'C', R1, -1.0
                  WRITE(40,1050), 'Y', I, J, K, 'C', R1 + (LF - NUMBU)
     &                                 *TC, -1.0
                  WRITE(40,1050), 'Y', I, J, K, 'C', R2, -T(I,J,K)
  320         CONTINUE
          R2 = R2 + LFMT
  330     CONTINUE
  340 CONTINUE
```

```
C      R = TCTT + LFMT + 2*LFMT*TC + 2*(LF - NUMBU)*TC + TC

C VARIABLE S(I,J):
      R = LFMT + TCTT + 2*LFMT*TC

C*****************************************************************************
C IF THE USER WANTS TO ALLOW A TARGET COMPLEX TO BE ATTACKED BY TWO LAUNCH
C FIELDS, ADD THE FOLLOWING ASSIGNMENT STATEMENT:
C       R2 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 3*TC
C     &      + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 5
C*****************************************************************************

C STEP THROUGH EACH LAUNCH FIELD:
      DO 370 I = 1, LF
          R1 = LFMT + TCTT + 2*LFMT*TC + 2*(LF - NUMBU)*TC

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 350 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS A BACKUP ONE, GO TO THE NEXT LAUNCH FIELD:
              IF (I .EQ. BU(P)) GO TO 370
  350     CONTINUE

C STEP THROUGH EACH TARGET COMPLEX.  S(I,J) WILL BE IN CONSTRAINT R WITH A
C COEFFICIENT OF 1.0, IN CONSTRAINT R + (LF - NUMBU)*TC WITH A COEFFICIENT OF
C THE NUMBER OF MISSILE TYPES, AND IN CONSTRAINT R1 WITH A COEFFICIENT OF 1.0:
          DO 360 J = 1, TC
                R = R + 1
                R1 = R1 + 1
                WRITE(40,1060), 'S', I, J, 'C', R, 1.0
                WRITE(40,1060), 'S', I, J, 'C', R + (LF - NUMBU)*TC,
     &                             REAL(MT)
                WRITE(40,1060), 'S', I, J, 'C', R1, 1.0

C*****************************************************************************
C IF THE USER WANTS TO ALLOW A TARGET COMPLEX TO BE ATTACKED BY TWO LAUNCH
C FIELDS, ADD THE FOLLOWING STATEMENT:
C                 WRITE(40,1060) 'S', I, J, 'C', R2 + 1, 1.0
C*****************************************************************************

  360     CONTINUE
  370 CONTINUE

C*****************************************************************************
C       M = LFMT (OR SOME OTHER LARGE ENOUGH NUMBER)
C       R = LFMT + TCTT
C       R1 = LFMT + TCTT + 2*LFMT*TC
C       R2 = LFMT + TCTT + 4*LFMT*TC
C       DO 340 I = 1, LF
C           DO 310 P = 1, NUMBU
C               IF (I .EQ. BU(P)) GO TO 340
```

```
C  310     CONTINUE
C          DO 330 J = 1, TC
C              R1 = R1 + 1
C              DO 320 K = 1, MT
C                  IF (MADJ(I,K) .EQ. 0) GO TO 320
C                  ROW(I,J,K) = R1
C  320         CONTINUE
C  330     CONTINUE
C  340 CONTINUE
C      DO 375 I = 1, LF
C          DO 350 P = 1, NUMBU
C              IF (I .EQ. BU(P)) GO TO 375
C  350     CONTINUE
C          R2 = R2 - LFMT*TC
C          R3 = 0
C          DO 370 J = 1, TC
C              R2 = R2 - R3
C              R3 = 0
C              DO 360 K = 1, MT
C                  IF (MADJ(I,K) .EQ. 0) GO TO 360
C                  R = R + 1
C                  R2 = R2 + 1
C                  R3 = R3 + 1
C                  WRITE(40,1050) 'Y', I, J, K, 'C', R, 1.0
C                  WRITE(40,1050) 'Y', I, J, K, 'C', R + LFMT*TC,
C     &                               REAL(NUMTAR(J))
C                  WRITE(40,1050) 'Y', I, J, K, 'C', ROW(I,J,K), REAL(M)
C                  DO 353 II = 1, LF
C                      DO 351 P = 1, NUMBU
C                          IF (II .EQ. BU(P)) GO TO 353
C  351                 CONTINUE
C                      IF (I .EQ. II) GO TO 353
C                      DO 352 KK = 1, MT
C                          IF (MADJ(II,KK) .EQ. 0) GO TO 352
C                          WRITE(40,1050) 'Y', I, J, K, 'C',
C     &                                       ROW(II,J,KK), 1.0
C  352                 CONTINUE
C  353             CONTINUE
C                  WRITE(40,1050) 'Y', I, J, K, 'C', R2, -T(I,J,K)
C  360         CONTINUE
C  370     CONTINUE
C  375 CONTINUE
C*****************************************************************************

C VARIABLE DELTA(J):
      R = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + TC
      R1 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 2*TC

C*****************************************************************************
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, COMMENT OUT THE ABOVE
C ASSIGNMENT STATEMENTS FOR R AND R1 AND INCLUDE THE BELOW ONES:
```

```
C      R = LFMT + TCTT + 4*LFMT*TC
C      R1 = R + TC

C*******************************************************************************

C STEP THROUGH EACH TARGET COMPLEX.  DELTA(J) WILL BE IN CONSTRAINT R WITH A
C COEFFICIENT OF TMAX AND IN CONSTRAINT R1 + 1 WITH A COEFFICIENT OF 1.0:
      DO 380 J = 1, TC
          R = R + 1
          WRITE(40,1065), 'DELTA', J, 'C', R, TMAX
          WRITE(40,1065), 'DELTA', J, 'C', R1 + 1, 1.0
  380 CONTINUE
      WRITE(40,1040), 'INT'

C VARIABLE Z(J):
      R1 = LFMT + TCTT + 2*LFMT*TC + 2*(LF - NUMBU)*TC + TC
      R2 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + TC
      R3 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC + 3
      R4 = 0
      R5 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC + 3
     &     + NUMGOOD*NUMFAIR

C*******************************************************************************
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, COMMENT OUT THE ABOVE
C ASSIGNMENT STATEMENTS FOR R1, R2, R3, R4, AND R5 AND INCLUDE THE BELOW ONES:

C      R1 = LFMT + TCTT + 3*LFMT*TC
C      R2 = R1 + LFMT*TC
C      R3 = R2 + 3*TC + 3
C      R4 = 0
C      R5 = R3 + NUMGOOD*NUMFAIR

C*******************************************************************************

C STEP THROUGH EACH TARGET COMPLEX:
      DO 440 J = 1, TC

C Z(J) WILL BE IN CONSTRAINTS R1 + 1 TO R1 + LFMT WITH COEFFICIENTS OF 1.0:
          DO 390 R = 1, LFMT
              WRITE(40,1070), 'Z', J, 'C', R1 + R, 1.0
  390     CONTINUE
          R1 = R1 + LFMT
          R2 = R2 + 1

C Z(J) WILL BE IN CONSTRAINT R2 WITH A COEFFICIENT OF -1.0:
          WRITE(40,1070), 'Z', J, 'C', R2, -1.0

C Z(J) WILL BE IN CONSTRAINT R2 + TC + 1 WITH A COEFFICIENT OF -1.0:
          WRITE(40,1070), 'Z', J, 'C', R2 + TC + 1, -1.0
```

```
C Z(J) WILL BE IN CONSTRAINT R2 + 2*TC + 1 WITH A COEFFICIENT OF -1.0:
          WRITE(40,1070), 'Z', J, 'C', R2 + 2*TC + 1, -1.0

C IF THERE ARE GOOD TARGETS AT COMPLEX J, THEN Z(J) WILL BE IN CONSTRAINTS
C R3 + 1 TO R3 + NUMFAIR WITH COEFFICIENTS OF 1.0:
          IF (NADJ(J,1) .GT. 0) THEN
              DO 400 R = 1, NUMFAIR
                  R3 = R3 + 1
                  WRITE(40,1070), 'Z', J, 'C', R3, 1.0
  400         CONTINUE

C ELSE, IF THERE ARE FAIR TARGETS AT COMPLEX J,
          ELSEIF (NADJ(J,2) .GT. 0) THEN
              R4 = R4 + 1
              R6 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC +3
              R7 = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC +3

C*****************************************************************************
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, COMMENT OUT THE ABOVE
C ASSIGNMENTS FOR R6 AND R7 AND INCLUDE THE BELOW ONES:

C             R6 = LFMT + TCTT + 4*LFMT*TC + 3*TC +3
C             R7 = R6

C*****************************************************************************

C THEN STEP THROUGH EACH COMPLEX WITH GOOD TARGETS:
              DO 410 R = 1, NUMGOOD

C Z(J) WILL BE IN CONSTRAINT R6 + R4 WITH A COEFFICIENT OF -1.0:
                  WRITE(40,1070), 'Z', J, 'C', R6 + R4, -1.0
                  R6 = R6 + NUMFAIR
  410         CONTINUE

C STEP THROUGH EACH COMPLEX WITH POOR TARGETS:
              DO 420 R = 1, NUMPOOR

C Z(J) WILL BE IN CONSTRAINT R7 + NUMGOOD*NUMFAIR + R4 WITH A COEFFICIENT
C OF 1.0:
                  WRITE(40,1070), 'Z', J, 'C', R7 + NUMGOOD*NUMFAIR +
     &                                    R4, 1.0
                  R7 = R7 + NUMFAIR
  420         CONTINUE

C ELSE, COMPLEX J HAS ONLY POOR TARGETS:
          ELSE

C STEP THROUGH EACH COMPLEX WITH FAIR TARGETS:
              DO 430 R = 1, NUMFAIR
                  R5 = R5 + 1
```

```
C Z(J) WILL BE IN CONSTRAINT R5 WITH A COEFFICIENT OF -1.0:
                  WRITE(40,1070), 'Z', J, 'C', R5, -1.0
  430         CONTINUE
          END IF
  440 CONTINUE

C VARIABLE TFIRST:
      R = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + TC

C******************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, COMMENT OUT THE ABOVE
C ASSIGNMENT STATEMENT FOR R AND INCLUDE THIS ONE:

C      R = LFMT + TCTT + 4*LFMT*TC

C******************************************************************************

C STEP THROUGH EACH TARGET COMPLEX.  TFIRST WILL BE IN CONSTRAINTS R + 1 TO
C R + TC WITH COEFFICIENTS OF 1.0:
      DO 450 J = 1, TC
          R = R + 1
          WRITE(40,1080), 'TFIRST', 'C', R, 1.0
  450 CONTINUE

      R = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 2*TC + 1

C STEP THROUGH EACH TARGET COMPLEX.  TFIRST WILL BE IN CONSTRAINTS R + 1 TO
C R + TC WITH COEFFICIENTS OF 1.0:
      DO 455 J = 1, TC
          R = R + 1
          WRITE(40,1080), 'TFIRST', 'C', R, 1.0
  455 CONTINUE

      R = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC + 2

C******************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, COMMENT OUT THE ABOVE
C ASSIGNMENT STATEMENT FOR R AND INCLUDE THIS ONE:

C      R = LFMT + TCTT + 4*LFMT*TC + 3*TC + 2

C******************************************************************************

      WRITE(40,1080), 'TFIRST', 'C', R, 1.0
      WRITE(40,1080), 'TFIRST', 'C', R + 1, -PCT
      WRITE(40,1080), 'TFIRST', 'C', R + 2 + NUMGOOD*NUMFAIR +
     &                NUMFAIR*NUMPOOR, 1.0

C VARIABLE TLAST:
      R = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 3*TC + 1
```

```
C****************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, COMMENT OUT THE ABOVE
C ASSIGNMENT STATEMENT FOR R AND INCLUDE THIS ONE:

C      R = LFMT + TCTT + 4*LFMT*TC + 2*TC + 1

C****************************************************************************

C STEP THROUGH EACH TARGET COMPLEX.  TLAST WILL BE IN CONSTRAINTS R + 1 TO
C R + TC WITH COEFFICIENTS OF 1.0:
      DO 460 J = 1, TC
          R = R + 1
          WRITE(40,1080), 'TLAST', 'C', R, 1.0
  460 CONTINUE
      WRITE(40,1080), 'TLAST', 'C', R + 1, -1.0

C VARIABLE DUR:
      WRITE(40,1080), 'DUR', 'C', R + 1, 1.0
      WRITE(40,1080), 'DUR', 'C', R + 2, 1.0
      R1 = R + 4 + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR
      WRITE(40,1080), 'DUR', 'C', R1, 1.0

C VARIABLE O1:
      WRITE(40,1090), 'O1', 'OBJ', W1
      WRITE(40,1095), 'O1', 'C', R1 - 1, -1.0

C VARIABLE O2:
      WRITE(40,1090), 'O2', 'OBJ', W2
      WRITE(40,1095), 'O2', 'C', R1, -1.0

C THE LAST SECTION LISTS THE RIGHT HAND SIDE VALUES FOR EACH CONSTRAINT:
      WRITE(40,1040), 'RHS'
      Q = 0

C STEP THROUGH EACH TARGET COMPLEX:
      DO 480 J = 1, TC

C STEP THROUGH EACH TARGET TYPE:
          DO 470 L = 1, TT

C IF THERE ARE NO TARGETS OF TYPE L AT COMPLEX J, GO TO THE NEXT TARGET TYPE:
              IF (NADJ(J,L) .EQ. 0) GO TO 470
              Q = Q + 1
              WRITE(40,1100), 'RHS', 'C', Q, REAL(NADJ(J,L))
  470     CONTINUE
  480 CONTINUE
      Q = TCTT

C STEP THROUGH EACH LAUNCH FIELD:
      DO 510 I = 1, LF
```

```
C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 490 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS A BACKUP ONE, GO TO THE NEXT LAUNCH FIELD:
              IF (I .EQ. BU(P)) GO TO 510
  490     CONTINUE

C STEP THROUGH EACH MISSILE TYPE:
          DO 500 K = 1, MT

C IF THERE ARE NO MISSILES OF TYPE K AT LAUNCH FIELD I, GO TO THE NEXT MISSILE
C TYPE:
              IF (MADJ(I,K) .EQ. 0) GO TO 500
              Q = Q + 1
              WRITE(40,1100), 'RHS', 'C', Q, REAL(MADJ(I,K))
  500     CONTINUE
  510 CONTINUE

C******************************************************************************
C IF THE USER DOES NOT WANT TO USE S(I,J) VARIABLES, THEN INCLUDE THE
C FOLLOWING SECTION:

C      Q = Q + 2*LFMT*TC
C      DO 518 I = 1, LF
C          DO 511 P = 1, NUMBU
C              IF (I .EQ. BU(P)) GO TO 518
C  511     CONTINUE
C          DO 517 J = 1, TC
C              DO 515 K = 1, MT
C                  IF (MADJ(I,K) .EQ. 0) GO TO 515
C                  Q = Q + 1
C                  WRITE(40,1100), 'RHS', 'C', Q, M
C  515         CONTINUE
C  517     CONTINUE
C  518 CONTINUE

C******************************************************************************

C******************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, COMMENT OUT THE
C FOLLOWING SECTION:

      Q = TCTT + LFMT + 2*LFMT*TC + 2*(LF - NUMBU)*TC

C      Q = Q +
C STEP THROUGH EACH TARGET COMPLEX.  THE RIGHT HAND SIDE OF CONSTRAINTS Q + 1
C TO Q + TC WILL BE 1.0:
      DO 520 J = 1, TC
          Q = Q + 1
          WRITE(40,1100), 'RHS', 'C', Q, 1.0
```

```
C*******************************************************************************
C IF THE USER WANTS TO ALLOW A TARGET COMPLEX TO BE ATTACKED BY TWO LAUNCH
C FIELDS, COMMENT OUT THIS STATEMENT AND INCLUDE THE NEXT:
            WRITE(40,1100), 'RHS', 'C', Q, 1.0
C            WRITE(40,1100) 'RHS', 'C', Q, 2.0
C*******************************************************************************

  520 CONTINUE

C*******************************************************************************

      Q = TCTT + LFMT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 2*TC + 1

C*******************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, COMMENT OUT THE
C ABOVE ASSIGNMENT STATEMENT FOR Q AND INCLUDE THIS ONE:

C      Q = TCTT + LFMT + 4*LFMT*TC + TC + 1

C*******************************************************************************

      WRITE(40,1100), 'RHS', 'C', Q, TC - 1.0

      Q = TCTT + LFMT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC + 3
     &    + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR

C*******************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, COMMENT OUT THE
C ABOVE ASSIGNMENT STATEMENT FOR Q AND INCLUDE THIS ONE:

C      Q = Q + 2*TC + 2 + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR

C*******************************************************************************

      WRITE(40,1100), 'RHS', 'C', Q + 1, TMIN

C*******************************************************************************
C IF THE USER WANTS TO ALLOW A TARGET COMPLEX TO BE ATTACKED BY TWO LAUNCH
C FIELDS, INCLUDE THESE STATEMENTS:
C      Q = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC
C     &    + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 5
C      WRITE(40,1100) 'RHS', 'C', Q + 1, TC + 1
C*******************************************************************************

C*******************************************************************************
C IF THE SOLUTION IS UNBALANCED, ADD THE FOLLOWING SECTION:
C      Q = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC
C     &    + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 6
C      DO 540 I = 1, LF
C          DO 530 P = 1, NUMBU
C              IF (I .EQ. BU(P)) GO TO 540
```

```
C  530     CONTINUE
C          Q = Q + 1
C          WRITE(40,1100) 'RHS', 'C', Q, NUMTAR(JSTAR)
C  540 CONTINUE
C*****************************************************************************

C*****************************************************************************
C IF THE USER DOES NOT WANT EACH LAUNCH FIELD TO HAVE A PCTBU BACKUP, ADD THE
C FOLLOWING STATEMENTS:
C        Q = LFMT + TCTT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC
C      &     + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 5
C        WRITE(40,1100) 'RHS', 'C', Q + 1, TOTMIS
C*****************************************************************************

      WRITE(40,1120), 'ENDATA'
      CLOSE(40)

C THIS SECTION CREATES THE SPECS FILE FOR ZOOM.

      OPEN(UNIT=50, FILE='SILOATT.SPC', ACCESS='SEQUENTIAL',
     &     STATUS='UNKNOWN', IOSTAT=IERROR, ERR=992)

      WRITE(50,1130), 'BEGIN', 'MINIMIZE'

      NUMROWS = TCTT + LFMT + 3*LFMT*TC + 2*(LF - NUMBU)*TC + 4*TC
     &          + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 5
      NUMVARS = LFMT*TCTT + LFMT*TC + (LF - NUMBU)*TC + 2*TC + 5
      NUMINT = LFMT*TC + (LF - NUMBU)*TC + TC

C*****************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, COMMENT OUT THE
C ABOVE ASSIGNMENT STATEMENTS FOR NUMROWS, NUMVARS, AND NUMINT AND INCLUDE
C THESE:

C       NUMROWS = TCTT + LFMT + 4*LFMT*TC + 3*TC
C     &           + NUMGOOD*NUMFAIR + NUMFAIR*NUMPOOR + 5
C       NUMVARS = LFMT*TCTT + LFMT*TC + 2*TC + 5
C       NUMINT = LFMT*TC + TC

C*****************************************************************************

      ITERLIM = 1000000
      MAXNODES = 100000
      MAXSAVE = 20
      WRITE(50,1140), 'ROWS', NUMROWS, 'COLUMNS', NUMVARS, 'BRANCH',
     &                'YES', 'BOUNDS', 'NONE', 'GAP', TMAX, 'INTEGER',
     &                NUMINT, 'LIMIT', ITERLIM, 'MAX NODES', MAXNODES,
     &                'MAX SAVE', MAXSAVE, 'OBJECTIVE', 'OBJ',
     &                'PRINT CONTINUOUS 0', 'PRINT LP1 0',
     &                'PRINT LP2 0', 'PRINT HEURISTIC 0',
     &                'PRINT BRANCH 0', 'PRINT TOUR 0', 'QUIT',
     )
     &                  'NO', 'END'
      CLOSE(50)
      RETURN

  991 OPEN(UNIT=15, FILE='SILOATT.ERR', ACCESS='SEQUENTIAL',
     &     STATUS='UNKNOWN')
      WRITE(15,*) 'ERROR IN OPENING FILE SILOATT.MPS'
      WRITE(15,*) 'ERROR CODE = ', IERROR
      CLOSE(15)
      STOP

  992 OPEN(UNIT=15, FILE='SILOATT.ERR', ACCESS='SEQUENTIAL',
     &     STATUS='UNKNOWN')
      WRITE(15,*) 'ERROR IN OPENING FILE SILOATT.SPC'
      WRITE(15,*) 'ERROR CODE = ', IERROR
      CLOSE(15)
      STOP

 1000 FORMAT(A4, 1X, A22, /, A4, /, 2X, A1, 1X, A3)
 1010 FORMAT(2X, A1, 1X, A1, I3)
 1020 FORMAT(2X, A1, 1X, A1, I3, /, 2X, A1, 1X, A1, I3)
 1030 FORMAT(4X, A1, I1, I1, I1, I1, 5X, A1, I3, 6X, F8.3)
 1040 FORMAT(A3)
 1050 FORMAT(4X, A1, I1, I1, I1, 6X, A1, I3, 6X, F8.3)
 1060 FORMAT(4X, A1, I1, I1, 7X, A1, I3, 6X, F8.3)
 1065 FORMAT(4X, A5, I1, 4X, A1, I3, 6X, F8.3)
 1070 FORMAT(4X, A1, I1, 8X, A1, I3, 6X, F8.3)
 1080 FORMAT(4X, A6, 4X, A1, I3, 6X, F8.3)
 1090 FORMAT(4X, A2, 8X, A3, 7X, F8.3)
 1095 FORMAT(4X, A2, 8X, A1, I3, 6X, F8.3)
 1100 FORMAT(4X, A3, 7X, A1, I3, 6X, F8.3)
 1110 FORMAT(A7)
 1120 FORMAT(A6)
 1130 FORMAT(A5, /, A8)
 1140 FORMAT(A4, 1X, I4, /, A7, 1X, I4, /, A6, 1X, A3, /,
     &       A6, 1X, A4, /, A3, 1X, F6.2, /, A7, 1X, I3, /,
     &       A5, 1X, I7, /, A9, 1X, I6, /, A8, 1X, I3, /,
     &       A9, 1x, A3, /, A18, /, A11, /, A11, /, A17, /, A14, /,
     &       A12, /, A4 , 1X, A2, /, A3)

      END

C*****************************************************************************
C*****************************************************************************
      SUBROUTINE PRNTSOL(BU, CPLX, FIELD, LF, MADJ, MAXLF, MAXMT,
     &                   MAXTC, MAXTT, MAXVAR, MT, MTYPE, NADJ,
     &                   NUMBU, T, TC, TT, TTYPE, VARVAL)
C******************************************************************************
C
C  NAME:  PRNTSOL
C
```

```
C  REV  DATE OF
C  NO.  CHANGE   DESIGNER     CCR  DESCRIPTION
C  ---  -------  ------------ ---- -------------------------------------------
C  000   9FEB93  R. PACE           ORIGINAL RELEASE
C
C  DESCRIPTION:  PRINTS THE SOLUTION INTO TWO OUTPUT FILES
C
C  CALLED BY:  SILOATT
C
C  CALLS:  NONE
C
C  CALLING SEQUENCE:
C
CIN  BU     - ARRAY STORING THE BACKUP LAUNCH FIELDS
C    CPLX   - ARRAY STORING THE NAMES OF THE TARGET COMPLEXES
C    FIELD  - ARRAY STORING THE NAMES OF THE LAUNCH FIELDS
c    LF     - NUMBER OF LAUNCH FIELDS
C    MADJ   - ARRAY STORING THE ADJUSTED NUMBER OF MISSILES BY TYPE AT EACH
C             LAUNCH FIELD
C    MAXLF  - MAX NUMBER OF LAUNCH FIELDS
C    MAXMT  - MAX NUMBER OF MISSILE TYPES
C    MAXTC  - MAX NUMBER OF TARGET COMPLEXES
C    MAXTT  - MAX NUMBER OF TARGET TYPES
C    MAXVAR - MAX NUMBER OF VARIABLES
C    MT     - NUMBER OF MISSILE TYPES
C    MTYPE  - ARRAY STORING THE NAMES OF THE MISSILE TYPES
C    NADJ   - ARRAY STORING THE ADJUSTED NUMBER OF TARGETS BY TYPE AT EACH
C             TARGET COMPLEX
C    NUMBU  - THE NUMBER OF BACKUP LAUNCH FIELDS
C    T      - ARRAY STORING FLIGHT TIMES OF MISSILES (BY TYPE) FROM EACH LAUNCH
C             FIELD TO EACH TARGET COMPLEX
C    TC     - NUMBER OF TARGET COMPLEXES
C    TT     - NUMBER OF TARGET TYPES
C    TTYPE  - ARRAY STORING THE NAMES OF THE TARGET TYPES
C    VARVAL - ARRAY STORING THE SOLUTION VECTOR
C
C FILES:
C
C    SILOATT.SOL
C    SINBAC.SOL
C    SILOATT.ERR
C
C LOCAL VARIABLES:
C
C    DELTA   - ARRAY STORING THE SOLUTION VALUES OF THE DELTA(J) VARIABLES
C    DUR     - STORES THE SOLUTION VALUE OF THE VARIABLE DUR
C    I, II, J, K, L, P, Q - COUNTERS
C    IERROR  - STORES THE VALUE OF IOSTAT
C    S       - ARRAY STORING THE SOLUTION VALUES OF THE S(I,J) VARIABLES
C    TFIRST  - STORES THE SOLUTION VALUE OF THE VARIABLE TFIRST
C    TLAST   - STORES THE SOLUTION VALUE OF THE VARIABLE TLAST
```

```
C    TOTMIS  - ARRAY STORING THE TOTAL NUMBER OF MISSILES OF TYPE K THAT ARE
C              ALLOCATED FROM LAUNCH FIELD I TO TARGET COMPLEX J
C    X       - ARRAY STORING THE SOLUTION VALUES OF THE X(I,J,K,L) VARIABLES
C    Y       - ARRAY STORING THE SOLUTION VALUES OF THE Y(I,J,K) VARIABLES
C    Z       - ARRAY STORING THE SOLUTION VALUES OF THE Z(J) VARIABLES
C
C  ERROR MESSAGES:  NONE
C
C*****************************************************************************
C
C  LOCAL VARIABLE DECLARATIONS:

      INTEGER  BU(MAXLF), DELTA(10), I, IERROR, II, J, K, L, LF,
     &         MADJ(MAXLF,MAXMT), MT, NADJ(MAXTC,MAXTT), NUMBU, P, Q,
     &         S(10,10), TC, TT, Y(10,10,5)

      REAL     DUR, T(MAXLF,MAXTC,MAXMT), TFIRST, TLAST,
     &         TOTMIS(10,10,5), X(10,10,5,5), Z(10)

      DOUBLE PRECISION VARVAL(MAXVAR)

      CHARACTER*8 CPLX(MAXTC), FIELD(MAXLF)

      CHARACTER*4 MTYPE(MAXMT), TTYPE(MAXTT)
C
C THIS SECTION ASSIGNS THE VALUES OF THE ZOOM VARIABLES INTO THE
C APPROPRIATE VARIABLES FOR THE MIP.
C
      Q = 0

C STEP THROUGH EACH LAUNCH FIELD:
      DO 50 I = 1, LF

C IF IT IS ONE OF THE BACKUP ONES, GO TO THE NEXT LAUNCH FIELD:
          DO 10 P = 1, NUMBU
              IF (I .EQ. BU(P)) GO TO 50
   10     CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
          DO 40 J = 1, TC

C STEP THROUGH EACH MISSILE TYPE:
              DO 30 K = 1, MT

C IF THERE ARE NO MISSILES OF TYPE K AT LAUNCH FIELD I, GO TO THE NEXT
C MISSILE TYPE:
                  IF (MADJ(I,K) .EQ. 0) GO TO 30

C STEP THROUGH EACH TARGET TYPE:
                  DO 20 L = 1, TT
```

```
C IF THERE ARE NO TARGETS OF TYPE L AT COMPLEX J, GO TO THE NEXT TARGET TYPE:
                      IF (NADJ(J,L) .EQ. 0) GO TO 20

C OTHERWISE, SET X(I,J,K,L) EQUAL TO THE NEXT VALUE IN THE SOLUTION VECTOR:
                      Q = Q + 1
                      X(I,J,K,L) = VARVAL(Q)
   20             CONTINUE
   30          CONTINUE
   40      CONTINUE
   50 CONTINUE

C STEP THROUGH EACH LAUNCH FIELD:
      DO 90 I = 1, LF

C IF IT IS ONE OF THE BACKUP ONES, GO TO THE NEXT LAUNCH FIELD:
          DO 60 P = 1, NUMBU
              IF (I .EQ. BU(P)) GO TO 90
   60     CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
          DO 80 J = 1, TC

C STEP THROUGH EACH MISSILE TYPE:
              DO 70 K = 1, MT

C IF THERE ARE NO MISSILES OF TYPE K AT LAUNCH FIELD I, GO TO THE NEXT
C MISSILE TYPE:
                  IF (MADJ(I,K) .EQ. 0) GO TO 70

C OTHERWISE, SET Y(I,J,K) EQUAL TO THE NEXT VALUE IN THE SOLUTION VECTOR:
                  Q = Q + 1
                  Y(I,J,K) = VARVAL(Q)
   70         CONTINUE
   80     CONTINUE
   90 CONTINUE

C*******************************************************************************
C IF THE USER DOES NOT WANT TO USE THE S(I,J) VARIABLES, THEN DO NOT DECLARE
C S(I,J) (SEE ABOVE) AND COMMENT OUT THE FOLLOWING SECTION:

C STEP THROUGH EACH LAUNCH FIELD:
      DO 120 I = 1, LF

C IF IT IS ONE OF THE BACKUP ONES, GO TO THE NEXT LAUNCH FIELD:
          DO 100 P = 1, NUMBU
              IF (I .EQ. BU(P)) GO TO 120
  100     CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
          DO 110 J = 1, TC
```

```
C SET S(I,J) EQUAL TO THE NEXT VALUE IN THE SOLUTION VECTOR:
              Q = Q + 1
              S(I,J) = VARVAL(Q)
  110     CONTINUE
  120 CONTINUE

C**************************************************************************

C STEP THROUGH TARGET COMPLEX:
      DO 130 J = 1, TC

C SET DELTA(J) EQUAL TO THE NEXT VALUE IN THE SOLUTION VECTOR:
          Q = Q + 1
          DELTA(J) = VARVAL(Q)
  130 CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
      DO 140 J = 1, TC

C SET Z(J) EQUAL TO THE NEXT VALUE IN THE SOLUTION VECTOR:
          Q = Q + 1
          Z(J) = VARVAL(Q)
  140 CONTINUE

C SET TFIRST, TLAST, AND DUR EQUAL TO THE NEXT THREE VALUES IN THE SOLUTION
C VECTOR:
      TFIRST = VARVAL(Q+1)
      TLAST = VARVAL(Q+2)
      DUR = VARVAL(Q+3)

C THIS SECTION PRODUCES THE OUTPUT FOR THE J-8 SOLUTION FILE:

      OPEN(UNIT=16, FILE='SILOATT.SOL', ACCESS='SEQUENTIAL',
     &     STATUS='UNKNOWN', IOSTAT=IERROR, ERR=991)
      WRITE(16,1000)

C STEP THROUGH EACH LAUNCH FIELD:
      DO 170 I = 1, LF

C STEP THROUGH EACH BACKUP LAUNCH FIELD:
          DO 160 P = 1, NUMBU

C IF THE CURRENT LAUNCH FIELD IS FOR BACKUP ONLY, THEN "SLIDE" THE NAME OF
C EACH SUBSEQUENT LAUNCH FIELD FORWARD IN THE LIST (THEREBY ERASING THE NAME
C OF THE BACKUP FIELD:
              IF (I .EQ. BU(P)) THEN
                  DO 150 II = I, LF
                      FIELD(II) = FIELD(II+1)
                      GO TO 170
  150             CONTINUE
              END IF
```

```
160     CONTINUE
170 CONTINUE
    LF = LF - NUMBU
    WRITE(16,1010) (FIELD(I), I=1,LF)

C STEP THROUGH EACH TARGET COMPLEX:
    DO 210 J = 1, TC

C STEP THROUGH EACH TARGET TYPE:
      DO 200 L = 1, TT

C IF THERE ARE NO TARGETS OF TYPE L AT COMPLEX J, GO TO THE NEXT TARGET TYPE:
        IF (NADJ(J,L) .EQ. 0) GO TO 200

C OTHERWISE, WRITE THE NAME OF THE CURRENT TARGET COMPLEX TO 'SILOATT.SOL',
C AND THEN STEP THROUGH EACH LAUNCH FIELD:
        WRITE(16,1020) CPLX(J)
        DO 190 I = 1, LF

C STEP THROUGH EACH MISSILE TYPE:
            DO 180 K = 1, MT

C IF NO MISSILES OF TYPE K HAVE BEEN ALLOCATED FROM LAUNCH FIELD I TO TYPE
C L TARGETS AT TARGET COMPLEX J, THEN GO TO THE NEXT MISSILE TYPE:
                IF (X(I,J,K,L) .EQ. 0) GO TO 180

C THIS IF STATEMENT BLOCK WRITES THE SOLUTION DATA UNDER THE COLUMN FOR THE
C APPROPRIATE LAUNCH FIELD:
                IF (I .EQ. 1) THEN
                  IF (Z(J) .GE. TFIRST) THEN
                    WRITE(16,1030) TTYPE(L), MTYPE(K),
     &                          INT(X(I,J,K,L)), Z(J)
                  ELSE
                    WRITE(16,1030) TTYPE(L), MTYPE(K),
     &                          INT(X(I,J,K,L)), TFIRST
                  END IF
                ELSE IF (I .EQ. 2) THEN
                  IF (Z(J) .GE. TFIRST) THEN
                    WRITE(16,1040) TTYPE(L), MTYPE(K),
     &                          INT(X(I,J,K,L)), Z(J)
                  ELSE
                    WRITE(16,1040) TTYPE(L), MTYPE(K),
     &                          INT(X(I,J,K,L)), TFIRST
                  END IF
                ELSE IF (I .EQ. 3) THEN
                  IF (Z(J) .GE. TFIRST) THEN
                    WRITE(16,1050) TTYPE(L), MTYPE(K),
     &                          INT(X(I,J,K,L)), Z(J)
                  ELSE
                    WRITE(16,1050) TTYPE(L), MTYPE(K),
     &                          INT(X(I,J,K,L)), TFIRST
```

```
                    END IF
                  ELSE
                    IF (Z(J) .GE. TFIRST) THEN
                      WRITE(16,1060) TTYPE(L), MTYPE(K),
     &                              INT(X(I,J,K,L)), Z(J)
                    ELSE
                      WRITE(16,1060) TTYPE(L), MTYPE(K),
     &                              INT(X(I,J,K,L)), TFIRST
                    END IF
                  END IF
  180           CONTINUE
  190       CONTINUE
  200     CONTINUE
  210 CONTINUE
C
C THIS SECTION PRODUCES THE OUTPUT FOR THE SINBAC SOLUTION FILE:
C
      OPEN(UNIT=17, FILE='SINBAC.SOL', ACCESS='SEQUENTIAL',
     &     STATUS='UNKNOWN', IOSTAT=IERROR, ERR=992)

C STEP THROUGH EACH LAUNCH FIELD:
      DO 240 I = 1, LF

C STEP THROUGH EACH TARGET COMPLEX:
          DO 230 J = 1, TC

C STEP THROUGH EACH MISSILE TYPE AND SET TOTMIS(I,J,K) EQUAL TO 0:
              DO 220 K = 1, MT
                  TOTMIS(I,J,K) = 0
  220         CONTINUE
  230     CONTINUE
  240 CONTINUE

C STEP THROUGH EACH TARGET COMPLEX:
      DO 280 J = 1, TC

C SUM UP THE TOTAL NUMBER OF MISSILES OF TYPE K ATTACKING COMPLEX J FROM
C LAUNCH FIELD I:

C STEP THROUGH EACH LAUNCH FIELD:
          DO 270 I = 1, LF

C STEP THROUGH EACH MISSILE TYPE:
              DO 260 K = 1, MT

C STEP THROUGH EACH TARGET TYPE:
                  DO 250 L = 1, TT

C INCREASE THE CURRENT VALUE OF TOTMIS(I,J,K) BY THE VALUE OF X(I,J,K,L):
                      TOTMIS(I,J,K) = TOTMIS(I,J,K) + X(I,J,K,L)
  250             CONTINUE
```

```
C IF THERE ARE MISSILES OF TYPE K ALLOCATED FROM LAUNCH FIELD I TO TARGET
C COMPLEX J, WRITE THE SOLUTION DATA TO 'SINBAC SOL':
                IF (TOTMIS(I,J,K) .GT. 0.0) THEN
                  IF (TOTMIS(I,J,K) .GE. 10) THEN
                    WRITE(17,1070) CPLX(J)(1:3), MTYPE(K),
     &                  FIELD(I)(1:3), INT(10*TOTMIS(I,J,K))
                  ELSE IF (TOTMIS(I,J,K) .GE. 1) THEN
                    WRITE(17,1071) CPLX(J)(1:3), MTYPE(K),
     &                  FIELD(I)(1:3), INT(10*TOTMIS(I,J,K))
                  ELSE
                    WRITE(17,1072) CPLX(J)(1:3), MTYPE(K),
     &                  FIELD(I)(1:3), INT(10*TOTMIS(I,J,K))
                  END IF
                END IF
  260         CONTINUE
  270     CONTINUE
  280 CONTINUE

      CLOSE(16)
      CLOSE(17)
      RETURN

  991 OPEN(UNIT=15, FILE='SILOATT.ERR', STATUS='UNKNOWN')
      WRITE(15,*) 'ERROR IN OPENING FILE SILOATT.SOL'
      WRITE(15,*) 'ERROR CODE = ', IERROR
      CLOSE(15)

  992 OPEN(UNIT=15, FILE='SILOATT.ERR', STATUS='UNKNOWN')
      WRITE(15,*) 'ERROR IN OPENING FILE SINBAC.SOL'
      WRITE(15,*) 'ERROR CODE = ', IERROR
      CLOSE(15)

 1000 FORMAT(33X, 'LAUNCH FIELDS', /, 'TGT FLDS', 20X,
     &       '(NUMBER/TYPE/IMPACT TIME)', /, 11X)
 1010 FORMAT(11X, 10(A8, 9X))
 1020 FORMAT(A8)
 1030 FORMAT(1X, A4, 6X, A4, '/', I3, '/', F6.3)
 1040 FORMAT(1X, A4, 23X, A4, '/', I3, '/', F6.3)
 1050 FORMAT(1X, A4, 40X, A4, '/', I3, '/', F6.3)
 1060 FORMAT(1X, A4, 57X, A4, '/', I3, '/', F6.3)
 1070 FORMAT(A3, A4, A3, I3)
 1071 FORMAT(A3, A4, A3, '0', I2)
 1072 FORMAT(A3, A4, A3, '00', I1)

      END
*****************************************************************************
```

Appendix C. Output Files

C.1 Solution Summary for J-8 Review

The following is the output file produced by the FORTRAN program which is in the format that J-8 wants so that they can review the solution:

```
                                      LAUNCH FIELDS
TGT FLDS                         (TYPE/NUMBER/IMPACT TIME)

           NEW BERN         DURHAM           RALEIGH          WILMINGT
MACON
 GOOD                                                         GOOD/  5/35.150
 GOOD                                                         FAIR/ 14/35.150
ATL/TYB
 GOOD      GOOD/  3/35.980
 GOOD      FAIR/ 15/35.980
ATL/TYB
 FAIR      GOOD/ 18/35.980
SAVANNAH
 GOOD                       FAIR/ 21/34.027
COLUMBUS
 FAIR                                        FAIR/ 12/37.430
BRUNSWIC
 GOOD                                                         GOOD/ 24/37.430
ATHENS
 POOR                                        FAIR/  2/37.430
```

C.2 Solution Summary for SINBAC

The following is the output file produced by the FORTRAN program which is in the format that SINBAC needs in order to analyze the solution:

```
MACGOODWIL050
MACFAIRWIL140
ATLGOODNEW210
ATLFAIRNEW150
SAVFAIRDUR210
COLFAIRRAL120
BRUGOODWIL240
ATHFAIRRAL020
```

Appendix D. GAMS Input File

The following is the GAMS input file that was used to solve the unclassified sample problem:

```
$OFFSYMXREF OFFSYMLIST

SETS
     I  launch fields    / N-BERN, DURHAM, RALEIGH, WILM /
     J  target fields    / MACON, ATL-TYB, SAVAN, COLUMB, BRUNS, ATHENS /
     K  missile types / GOOD, FAIR /
     K2(K) missile types / GOOD, FAIR /
     L  target types     / GOOD, FAIR, POOR /
     G(J) fields with good tgts  / MACON, ATL-TYB, SAVAN, BRUNS /
     F(J) fields with fair tgts  / COLUMB /
     P(J) fields with poor tgts  / ATHENS /;

TABLE T(I,K,J)  flight times

                  MACON   ATL-TYB  SAVAN  COLUMB  BRUNS   ATHENS
N-BERN.GOOD       34.16   35.98    33.89   35.33  36.49    38.22
N-BERN.FAIR       32.61   33.24    32.66   33.10  33.61    34.82
DURHAM.GOOD        0       0        0       0      0        0
DURHAM.FAIR       32.11   32.405   32.45   32.51  32.96    34.71
RALEIGH.GOOD       0       0        0       0      0        0
RALEIGH.FAIR      32.95   33.475   33.09   33.42  33.92    35.34
WILM.GOOD         35.15   34.84    35.60   36.26  37.43    40.78
WILM.FAIR         32.30   32.565   32.67   32.70  33.14    34.94;

TABLE M(I,K)  number of missiles of type K at field I

           GOOD  FAIR
N-BERN      21    15
DURHAM       0    22
RALEIGH      0    15
WILM        29    17;

TABLE N(J,L)  number of targets of type L at field J

          GOOD  FAIR  POOR
MACON      19     0     0
```

```
ATL-TYB    18    18    0
SAVAN      21     0    0
COLUMB      0    12    0
BRUNS      24     0    0
ATHENS      0     0    2 ;


SCALARS  TMAX     maximum flight time possible  /40.78/
         TMIN     minimum flight time possible  /32.11/
         PCT      user-specified percentage     /1.0/
         TF       number of target fields       /6/
         Q        most launch flds that can hit each tgt fld   /1/
         MISSNUM  the number of missile types   /2/
         W1       relative weight of obj 1   /10/
         W2       relative weight of obj 2   /1/ ;

VARIABLES
     X(I,J,K,L)  num of msls of type k from i to tgts of type l at j
     Y(I,J,K)    indicator variable for if msls of type k from i to j
     S(I,J)      indicator variable for if msls allocated from i to j
     Z(J)        largest flight time of missiles allocated to j
     DEL(J)      indicator for disjunctive constraints
     TFIRST      flight time of the missile that impacts first
     TLAST       flight time of the missile that impacts last
     DUR         duration of the attack (tlast - tfirst)
     OPT         the optimal solution
     S1PLUS      the amount that obj 1 is over its goal
     S2PLUS      the amount that obj 2 is over its goal
     S1MINUS     the amount that obj 1 is under its goal
     S2MINUS     the amount that obj 2 is under its goal ;

FREE VARIABLE  OPT ;

POSITIVE VARIABLES  X, Z, TFIRST, TLAST, DUR, S1PLUS, S2PLUS, S1MINUS,
                    S2MINUS ;

BINARY VARIABLES  S, Y, DEL ;

EQUATIONS
     FLTTIME        objective function
     OBJ1           goal for tfirst
     OBJ2           goal for dur
     DEMAND(J,L)    attack each target
     SUPPLY(I,K)    cannot shoot more than available
     IND1(I,J,K)    set yijk to 1 if missiles of type k are
     IND2(I,J,K)       allocated from i to j
     IND3(I,J)      set sij to 1 if missiles allocated from i to j,
     IND4(I,J)         and to 0 otherwise
     MAXFT(I,J,K)   identify the max flt time into j
     DISJUN1(J)     disjunctive constraints to find the minimum
     DISJUN2           flight time
```

```
          TFIR(J)        ensure tfirst is set to the smallest z(j)
          TLAS(J)        find the maximum flight time
          DUR1           duration equals tlast minus tfirst
          DUR2           duration must be .le. a user-specified percent of tfirst
          MAXHIT(J)      do not attack each tgt field with more than 1 launch fld
          PRTY1(G,F)     attack good tgts before fair ones
          PRTY2(F,P)     attack fair tgts before poor ones;

FLTTIME ..  OPT =E= W1*S1PLUS + W2*S2PLUS ;
OBJ1 ..  TFIRST + S1MINUS - S1PLUS =E= TMIN ;
OBJ2 ..  DUR + S2MINUS - S2PLUS =E= 0 ;
DEMAND(J,L)$(N(J,L)) ..  SUM((I,K)$(M(I,K)), X(I,J,K,L)) =E= N(J,L) ;
SUPPLY(I,K)$(M(I,K)) ..  SUM((J,L)$(N(J,L)), X(I,J,K,L)) =L= M(I,K) ;
IND1(I,J,K)$(T(I,K,J)) ..  Y(I,J,K) =L= SUM(L$(N(J,L)), X(I,J,K,L)) ;
IND2(I,J,K)$(T(I,K,J)) .. Y(I,J,K) =G= (SUM(L$(N(J,L)),X(I.J,K,L)))/
(SUM(L,N(J,L))) ;
IND3(I,J) ..  S(I,J) =L= SUM(K$(M(I,K)), Y(I,J,K)) ;
IND4(I,J) ..  S(I,J) =G= (SUM(K$(M(I,K)), Y(I,J,K)))/MISSNUM ;
MAXFT(I,J,K)$(T(I,K,J)) ..  Z(J) =G= Y(I.J,K)*T(I,K,J) ;
DISJUN1(J) ..  TFIRST =G= Z(J) + DEL(J)*(-TMAX) ;
DISJUN2 ..  SUM(J, DEL(J)) =E= TF - 1 ;
TFIR(J) ..  TFIRST =L= Z(J) ;
TLAS(J) ..  TLAST =G= Z(J) ;
DUR1 ..  DUR =E= TLAST - TFIRST ;
DUR2 ..  DUR =L= PCT*TFIRST ;
MAXHIT(J) ..  SUM(I, S(I,J)) =E= Q ;
PRTY1(G,F) ..  Z(G) =L= Z(F) ;
PRTY2(F,P) ..  Z(F) =L= Z(P) ;

MODEL ALLOCATE /ALL/;

OPTION ITERLIM = 1000000 ;
OPTION RESLIM = 50000 ;
OPTION WORK = 100000 ;
OPTION OPTCR = 0.01 ;

OPTION LIMROW = 0 ;
OPTION LIMCOL = 0 ;

SOLVE ALLOCATE USING MIP MINIMIZING OPT ;
```

Appendix E. Complete Solutions and Allocation Summaries

Case 1: See chapter IV, section 4.3, p. 41.

Case 2: $Pct = 0.05$, $W_1 = 10$, $W_2 = 1$

SOLUTION:

Objective Function Value $= 37.159$

T_{first}	=	35.648	T_{last}	=	37.43	Dur	=	1.782
O_1	=	3.538	O_2	=	1.782			
X_{1211}	=	3	X_{1212}	=	18	X_{1221}	=	15
X_{2121}	=	19	X_{2623}	=	2	X_{3422}	=	12
X_{4311}	=	21	X_{4511}	=	8	X_{4521}	=	16
Y_{121}	=	1	Y_{122}	=	1	Y_{212}	=	1
Y_{262}	=	1	Y_{342}	=	1	Y_{431}	=	1
Y_{451}	=	1	Y_{452}	=	1			
S_{12}	=	1	S_{21}	=	1	S_{26}	=	1
S_{34}	=	1	S_{43}	=	1	S_{45}	=	1
Z_1	=	37.43	Z_2	=	35.98	Z_3	=	35.648
Z_4	=	37.43	Z_5	=	37.43	Z_6	=	37.43
δ_1	=	1	δ_2	=	1	δ_4	=	1
δ_5	=	1	δ_6	=	1			

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 8. Case 2 Allocation Summary

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX / TARGET TYPE Macon	Atl/Tyb		Savannah	Columbus	Bcaseswick	Athens
		good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair	19						2
Raleigh	fair					12		
Wilmington	good				21		8	
	fair						16	
Charlotte								
Lumberton								

Case 3: $Pct = 0.1$, $W_1 = 2$, $W_2 = 1$

SOLUTION:

Objective Function Value $= 7.237$

T_{first}	$= 34.027$	T_{last}	$= 37.43$	Dur	$= 3.403$
O_1	$= 1.917$	O_2	$= 3.403$		
X_{1211}	$= 3$	X_{1212}	$= 18$	X_{1221}	$= 15$
X_{2321}	$= 21$	X_{3422}	$= 12$	X_{4111}	$= 19$
X_{4511}	$= 10$	X_{4521}	$= 14$	X_{4623}	$= 2$
Y_{121}	$= 1$	Y_{122}	$= 1$	Y_{232}	$= 1$
Y_{342}	$= 1$	Y_{411}	$= 1$	Y_{451}	$= 1$
Y_{452}	$= 1$	Y_{462}	$=$		
S_{12}	$= 1$	S_{23}	$= 1$	S_{34}	$= 1$
S_{41}	$= 1$	S_{45}	$= 1$	S_{46}	$= 1$
Z_1	$= 35.15$	Z_2	$= 35.98$	Z_3	$= 34.027$
Z_4	$= 37.43$	Z_5	$= 37.43$	Z_6	$= 37.43$
δ_1	$= 1$	δ_2	$= 1$	δ_4	$= 1$
δ_5	$= 1$	δ_6	$= 1$		

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 9. Case 3 Allocation Summary

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX TARGET TYPE						
		Macon	Atl/Tyb		Savannah	Columbus	Bcaseswick	Athens
		good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair				21			
Raleigh	fair					12		
Wilmington	good	19					10	
	fair						14	2
Charlotte								
Lumberton								

Case 4: $Pct = 0.1$, $W_1 = 1$, $W_2 = 2$

SOLUTION:

Objective Function Value $= 5.32$

T_{first}	$=$	37.43	T_{last}	$=$	37.43	Dur	$=$	0
O_1	$=$	5.32	O_2	$=$	0			
X_{1211}	$=$	3	X_{1212}	$=$	18	X_{1221}	$=$	15
X_{2121}	$=$	19	X_{2623}	$=$	2	X_{3422}	$=$	12
X_{4311}	$=$	21	X_{4511}	$=$	8	X_{4521}	$=$	16
Y_{121}	$=$	1	Y_{122}	$=$	1	Y_{212}	$=$	1
Y_{262}	$=$	1	Y_{342}	$=$	1	Y_{431}	$=$	1
Y_{451}	$=$	1	Y_{452}	$=$	1			
S_{12}	$=$	1	S_{21}	$=$	1	S_{26}	$=$	1
S_{34}	$=$	1	S_{43}	$=$	1	S_{45}	$=$	1
Z_1	$=$	37.43	Z_2	$=$	37.43	Z_3	$=$	37.43
Z_4	$=$	37.43	Z_5	$=$	37.43	Z_6	$=$	37.43
δ_1	$=$	1	δ_2	$=$	1	δ_3	$=$	1
δ_4	$=$	1	δ_5	$=$	1			

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 10. Case 4 Allocation Summary

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX TARGET TYPE						
		Macon	Atl/Tyb		Savannah	Columbus	Bcaseswick	Athens
		good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair	19						2
Raleigh	fair					12		
Wilmington	good				21		8	
	fair						16	
Charlotte								
Lumberton								

Case 5: $Pct = 0.05$, $W_1 = 2$, $W_2 = 1$

SOLUTION:

Objective Function Value $= 8.858$

T_{first}	=	35.648	T_{last}	=	37.43	Dur	=	1.782
O_1	=	3.538	O_2	=	1.782			
X_{1211}	=	3	X_{1212}	=	18	X_{1221}	=	15
X_{2121}	=	19	X_{2623}	=	2	X_{3422}	=	12
X_{4311}	=	5	X_{4321}	=	16	X_{4511}	=	24
Y_{121}	=	1	Y_{122}	=	1	Y_{212}	=	1
Y_{262}	=	1	Y_{342}	=	1	Y_{431}	=	1
Y_{432}	=	1	Y_{451}	=	1			
S_{12}	=	1	S_{21}	=	1	S_{26}	=	1
S_{34}	=	1	S_{43}	=	1	S_{45}	=	1
Z_1	=	35.648	Z_2	=	35.98	Z_3	=	35.648
Z_4	=	37.43	Z_5	=	37.43	Z_6	=	37.43
δ_2	=	1	δ_3	=	1	δ_4	=	1
δ_5	=	1	δ_6	=	1			

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 11. Case 5 Allocation Summary

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX TARGET TYPE						
		Macon	Atl/Tyb		Savannah	Columbus	Bcaseswick	Athens
		good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair	19						2
Raleigh	fair					12		
Wilmington	good				5		24	
	fair				16			
Charlotte								
Lumberton								

Case 6: $Pct = 0.05$, $W_1 = 1$, $W_2 = 2$

SOLUTION:

Objective Function Value $= 5.32$

T_{first}	$=$	37.43	T_{last}	$=$	37.43	Dur	$=$	0
O_1	$=$	5.32	O_2	$=$	0			
X_{1211}	$=$	3	X_{1212}	$=$	18	X_{1221}	$=$	15
X_{2321}	$=$	21	X_{3422}	$=$	12	X_{4111}	$=$	5
X_{4121}	$=$	14	X_{4511}	$=$	24	X_{4623}	$=$	2
Y_{121}	$=$	1	Y_{122}	$=$	1	Y_{232}	$=$	1
Y_{342}	$=$	1	Y_{411}	$=$	1	Y_{412}	$=$	1
Y_{451}	$=$	1	Y_{462}	$=$	1			
S_{12}	$=$	1	S_{23}	$=$	1	S_{34}	$=$	1
S_{41}	$=$	1	S_{45}	$=$	1	S_{46}	$=$	1
Z_1	$=$	37.43	Z_2	$=$	37.43	Z_3	$=$	37.43
Z_4	$=$	37.43	Z_5	$=$	37.43	Z_6	$=$	37.43
δ_1	$=$	1	δ_2	$=$	1	δ_3	$=$	1
δ_4	$=$	1	δ_5	$=$	1			

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 12. Case 6 Allocation Summary

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX TARGET TYPE						
		Macon	Atl/Tyb		Savannah	Columbus	Braseswick	Athens
		good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair				21			
Raleigh	fair					12		
Wilmington	good	5					24	
	fair	14						2
Charlotte								
Lumberton								

Case 7: $Pct = 0.1$, $W_1 = 1$, $W_2 = 1$

SOLUTION:

Objective Function Value $= 5.32$

T_{first}	$=$	37.43	T_{last}	$=$	37.43	Dur	$=$	0
O_1	$=$	5.32	O_2	$=$	0			
X_{1211}	$=$	3	X_{1212}	$=$	18	X_{1221}	$=$	15
X_{2121}	$=$	19	X_{3422}	$=$	12	X_{3623}	$=$	2
X_{4311}	$=$	21	X_{4511}	$=$	8	X_{4521}	$=$	16
Y_{121}	$=$	1	Y_{122}	$=$	1	Y_{212}	$=$	1
Y_{342}	$=$	1	Y_{362}	$=$	1	Y_{431}	$=$	1
Y_{451}	$=$	1	Y_{452}	$=$	1			
S_{12}	$=$	1	S_{21}	$=$	1	S_{34}	$=$	1
S_{36}	$=$	1	S_{43}	$=$	1	S_{45}	$=$	1
Z_1	$=$	37.43	Z_2	$=$	37.43	Z_3	$=$	37.43
Z_4	$=$	37.43	Z_5	$=$	37.43	Z_6	$=$	37.43
δ_1	$=$	1	δ_2	$=$	1	δ_3	$=$	1
δ_4	$=$	1	δ_5	$=$	1			

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 13. Case 7 Allocation Summary

LAUNCH	MISSILE	TARGET COMPLEX / TARGET TYPE						
		Macon	Atl/Tyb		Savannah	Columbus	Bcaseswick	Athens
FIELD	TYPE	good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair	19						
Raleigh	fair					12		2
Wilmington	good				21		8	
	fair						16	
Charlotte								
Lumberton								

Case 8: $Pct = 0.05$, $W_1 = 1$, $W_2 = 1$

SOLUTION:

Objective Function Value $= 5.32$

T_{first}	=	35.648	T_{last}	=	37.43	Dur	=	1.782
O_1	=	3.538	O_2	=	1.782			
X_{1211}	=	3	X_{1212}	=	18	X_{1221}	=	15
X_{2321}	=	21	X_{3422}	=	12	X_{3623}	=	2
X_{4111}	=	5	X_{4121}	=	14	X_{4511}	=	24
Y_{121}	=	1	Y_{122}	=	1	Y_{232}	=	1
Y_{342}	=	1	Y_{362}	=	1	Y_{411}	=	1
Y_{412}	=	1	Y_{451}	=	1			
S_{12}	=	1	S_{23}	=	1	S_{34}	=	1
S_{36}	=	1	S_{41}	=	1	S_{45}	=	1
Z_1	=	35.648	Z_2	=	35.98	Z_3	=	35.648
Z_4	=	37.43	Z_5	=	37.43	Z_6	=	37.43
δ_2	=	1	δ_3	=	1	δ_4	=	1
δ_5	=	1	δ_6	=	1			

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 14. Case 8 Allocation Summary

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX / TARGET TYPE						
		Macon	Atl/Tyb		Savannah	Columbus	Bcaseswick	Athens
		good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair				21			
Raleigh	fair					12		2
Wilmington	good	5					24	
	fair	14						
Charlotte								
Lumberton								

Case 9: $Pct = 0.15$, $W_1 = 10$, $W_2 = 1$

SOLUTION:

Objective Function Value $= 9.26$

T_{first}	$= 32.548$	T_{last}	$= 37.43$	Dur	$= 4.882$
O_1	$= 0.438$	O_2	$= 4.882$		

X_{1211}	$= 3$	X_{1212}	$= 18$	X_{1221}	$= 15$
X_{2321}	$= 21$	X_{3422}	$= 12$	X_{4111}	$= 19$
X_{4511}	$= 10$	X_{4521}	$= 14$	X_{4623}	$= 2$

Y_{121}	$= 1$	Y_{122}	$= 1$	Y_{232}	$= 1$
Y_{342}	$= 1$	Y_{411}	$= 1$	Y_{451}	$= 1$
Y_{452}	$= 1$	Y_{462}	$= 1$		

S_{12}	$= 1$	S_{23}	$= 1$	S_{34}	$= 1$
S_{41}	$= 1$	S_{45}	$= 1$	S_{46}	$= 1$

Z_1	$= 35.15$	Z_2	$= 35.98$	Z_3	$= 32.548$
Z_4	$= 37.43$	Z_5	$= 37.43$	Z_6	$= 37.43$

δ_1	$= 1$	δ_2	$= 1$	δ_4	$= 1$
δ_5	$= 1$	δ_6	$= 1$		

all other variables $= 0$.

ALLOCATION SUMMARY:

Table 15. Case 9 Allocation Summary

LAUNCH FIELD	MISSILE TYPE	TARGET COMPLEX TARGET TYPE						
		Macon	Atl/Tyb		Savannah	Columbus	Bcaseswick	Athens
		good	good	fair	good	fair	good	poor
New Bern	good		3	18				
	fair		15					
Durham	fair				21			
Raleigh	tair					12		
Wilmington	good	19					10	
	fair						14	2
Charlotte								
Lumberton								

Appendix F. User's Guide

F.1 Introduction

This appendix contains instructions for using the FORTRAN program to solve a missile allocation problem.

F.2 How to Use SILOATT

1. Make sure that the files SILOATT.EXE and SILOATT.DAT are loaded in the current directory on the VAX/VMS computer system.
2. Edit the file SILOATT.DAT:
   (a) Input the data for number of missiles by type at each launch field into the first table following these instructions:
      i. Each row of this table must have the following format: launch field name (columns 1–8), one blank space, missile type (columns 10–13), four blank spaces, and number of missiles. The name of the launch field must start in column 1 and the missile type must start in column 10. If the name of a launch field (or missile type) is less than eight (four) characters, put in enough blanks to make the entire field eight (four) characters. If the name of a launch field (or missile type) is more than eight (four) characters, input no more than eight (four) characters. The number of missiles may begin in column 18 or beyond.

         Example:
         ```
         New Bern good    12
         New Bern fair    22
         Durham   fair    33
         Lumberto good    20
         ```
      ii. The first launch field listed will be considered by the program as launch field 1 from then on, the second one listed will be considered

launch field 2, and so on. In the example above, New Bern is launch field 1, Durham is launch field 2, and so on.

iii. Similarly, the first missile type listed will be considered missile type 1, the second one listed as missile type 2, and so on. In the example above, good is missile type 1 and fair is missile type 2.

iv. The numbers of missiles entered must be integer values only.

(b) Input the data for number of targets by type at each target complex into the second table following these instructions:

i. Each row of this table must have the following format: target complex name (columns 1–8), one blank space, target type (columns 10–13), four blank spaces, and number of targets. The name of the target complex must begin in column 1 and the target type must begin in column 13. If the name of a target complex (or target type) is less than eight (four) characters, put in enough blanks to make the entire field eight (four) characters. If the name of a target complex (or target type) is more than eight (four) characters, input no more than eight (four) characters. The number of targets may begin in column 18 or beyond.

Example:
```
Macon    good    120
Macon    fair    220
Savannah fair    330
Athens   good     20
```

ii. The first target complex listed will be considered by the program as target complex 1 from then on, the second one listed will be considered target complex 2, and so on. In the example above, Macon is target complex 1, Savannah is target complex 2, and so on.

iii. Similarly, the first target type listed will be considered target type 1, the second one listed as target type 2, and so on. In the example above, good is target type 1 and fair is target type 2.

iv. The numbers of targets entered must be integer values only.

(c) Input the data for the flight times of each missile type from each launch field to each target complex into the third table by following these instructions:

i. The first row of this table must contain the names of the target complexes. Input the names starting in column 18 (leave the rows with the headings in please, including the headings "LAUNCH FIELD" and "MISS TYPE"). Each target complex name should be eight characters long followed by one blank space. These names must appear *exactly* as in the second table *and* be in the same order (but list each target complex only once here).

ii. Leave one blank line and then type in the rest of the rows in the following format: launch field name (columns 1–8), one blank space, missile type (columns 10–13), four blank spaces, and the flight time of the appropriate type of missile from the current launch field to each target complex. The names of the launch fields must begin in column 1 and the missile types must begin in column 10. The flight times should be no larger than 999.99 and should be rounded off to the nearest one-hundredth of a minute. Each flight time entry should start directly under the first character of the appropriate target complex and should take up six spaces followed by three blank spaces.

Example:
```
LAUNCH   MISS
FIELD    TYPE    MACON    SAVANNAH ATHENS

New Bern good    34.55    35.11    39.99
New Bern fair    35.66    36.92    40.82
```

```
Durham   fair    47.33    43.22    45.99
Lumberto good    37.82    32.94    38.39
```

iii. Note that each launch field and each miss. type must appear *exactly* the same as it does in the first table *and* be in the exact same order.

(d) Input the value of the weighting factors for minimizing the earliest flight time (W_1) and duration (W_2). These must be decimal form (real numbers).

(e) Input the percentage of the earliest flight time that the duration is allowed to be (*Pct*). This must be in decimal form (i.e., 0.1 instead of 10%).

(f) Input the percent backup (*Pctbu*) that is required to be in each launch field. This must also be in decimal form.

(g) Indicate whether any of the launch fields should be reserved for backup only by typing 'Y' or 'y' for yes and 'N' or 'n' for no. If there are such launch fields, indicate how many and which ones (list by number, not by name).

(h) Indicate whether any of the target complexes should be combined by typing 'Y' or 'y' for yes and 'N' or 'n' for no. If any should be combined, list which ones (smallest number first) and the name of the combined complex (maximum of eight characters).

3. At the command prompt, type

```
def for029 siloatt.spc
def for024 siloatt.mps
```

which tells the computer that the files `siloatt.spc` and `siloatt.mps` should be equated to FORTRAN units 29 and 24 respectively. This needs to be done because ZOOM accesses these units directly without using OPEN and CLOSE statements.

4. At the command prompt, type

```
run siloatt
```

which causes the program to execute.

5. The program creates the following output files:

SILOATT.SOL	-	Contains the allocation summary.
SILOATT.ERR	-	Contains appropriate error messages. Only created if
	-	there was a problem encountered during execution.
SINBAC.SOL	-	Contains the allocation in the SINBAC format.
FOR025.DAT	-	Created by ZOOM. Contains the entire solution to the
	-	mixed-integer program.

6. If a feasible solution is found, the program creates the above mentioned .SOL files. If this solution is unsatisfactory, the user may want to modify the solution directly (just make sure the new solution is still feasible). Otherwise, the user could either modify the input data and re-run the program or modify a constraint by changing something in the FORTRAN code. If the FORTRAN code is changed, be sure to re-compile the entire program (including all the ZOOM files) before trying to re-run it. Also, make sure to record the changes made so that they can be un-made later!

7. If no feasible solution exists, an error message stating so is written in file SILOATT.ERR. At this point, the user could again choose either to modify the input data file or the FORTRAN code and re-run the program.

Bibliography

Bozovich, J. F., P. L. Groover, P. C. Ryan, and J. A. Battilega. *Mathematical Formulation of the Arsenal Exchange Model.* Denver: Martin Marietta Corporation, 1973.

Bracken, Jerome and James T. McGill. "A Convex Programming Model for Optimizing SLBM Attack of Bomber Bases," *Operations Research*, 21: 30–36 (January-February 1973).

Bunnell, Capt Robert E. and Capt Richard A. Takacs. *BRIK: An Interactive, Goal Programming Model for Nuclear Exchange Problems.* MS Thesis, AFIT/GST/OS/84M-5. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB (AU), OH, March 1984 (AD-A141092).

Davidson, CPT Peter A. Personal Correspondence. Defense Information Systems Agency, Defense Systems Support Organization, Pentagon, Washington D.C., 18 May 1992.

-----. Personal Correspondence. Defense Information Systems Agency, Defense Systems Support Organization, Pentagon, Washington D.C., 17 June 1992.

Day, Richard H. "Allocating Weapons to Target Complexes by Means of Nonlinear Programming," *Operations Research*, 14: 992–1013 (November-December 1966).

Den Broeder, G. G., Jr., R. E. Elison, and L. Emerling. "On Optimum Target Assignments," *Operations Research*, 7: 322–326 (March-April 1959).

Garfinkel, Robert S. and George L. Nemhauser. *Integer Programming.* New York: John Wiley & Sons, Inc., 1972.

Grotte, Jeffrey H. "An Optimizing Nuclear Exchange Model for the Analysis of Nuclear War and Deterrence," *Operations Research*, 30: 428–445 (May-June 1982).

Hillerman, Neal H. *The Theoretical Basis of the CODE 50 Nuclear Exchange Model.* Arlington, VA: Center for Naval Analysis, Research Contribution 173: September 1971 (AD-A043377).

Lemus, F. and K. H. David. "An Optimum Allocation of Different Weapons to a Target Complex," *Operations Research*, 11: 787–794 (September-October 1963).

Matlin, Samuel. "A Review of the Literature on the Missile-Allocation Problem," *Operations Research*, 18: 334–373 (March-April 1970).

Seiler, George J. *Strategic Nuclear Force Requirements and Issues.* Maxwell AFB, AL: Air University Press, February 1983 (RN AU-ARI-82-1).

Wambsganss, Capt Michael C. *A Linear Programming Multiple Objective Approach to Nuclear Exchange Modeling.* MS Thesis, AFIT/GOR/OS/82D-11. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1982 (AD-A135658).

Winston, Wayne L. *Introduction to Mathematical Programming: Applications and Algorithms.* Boston: PWS Kent Publishing Company, 1991.

Vita

Captain Richard C. Pace was born on 29 July 1963 in New Orleans, Louisiana. He graduated from Champaign Central High School in Champaign, Illinois in 1981 and attended Wheaton College in Wheaton, Illinois, graduating with a Bachelor of Arts in Mathematics (with an emphasis in Computer Science) in May 1985. Upon graduation, he received a reserve commission in the United States Army and served his first tour of duty at Fort Sill, Oklahoma. He began as an Assistant S-3/Chemical Officer for the 4th Battalion, 4th Field Artillery where he became the battalion's Special Weapons Officer, responsible for all nuclear weapons training in the battalion. In June 1988, he became the Assistant Chemical Officer for the 75th Field Artillery Brigade, also at Fort Sill. He stayed there, assisting the brigade in their nuclear, biological, and chemical (NBC) defense training until July 1989 when he was selected to attend the chemical officer advanced course at Fort McClellan, Alabama. Upon graduation, he was appointed the Instructor for the Radiological Operations portion of the chemical officer advanced course. He instructed at the Chemical School until entering the School of Engineering, Air Force Institute of Technology, in August 1991.

Permanent address: 3332 63rd Street
Woodridge, Illinois 60517

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE March 1993	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
OPTIMAL SILO ATTACK PLAN FOR THE RED INTEGRATED STRATEGIC OFFENSIVE PLAN (RISOP)

5. FUNDING NUMBERS

6. AUTHOR(S)
Richard C. Pace, Captain, US Army

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/93M-15

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Joint Staff Directorate for Force Structure, Resources, and Assessment (J-8)
Pentagon, Washington, D.C. 20301

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Joint Staff Directorate for Force Structure, Resources, and Assessment (J-8) sought a procedure which could be used to generate an optimal missile allocation for the silo attack portion of the Red Integrated Strategic Offensive Plan (RISOP). Their current solution procedure is a manual heuristic which is time-consuming and is not guaranteed to lead to an optimal solution. J-8 defines an optimal solution as a feasible solution which minimizes both the flight time of the missile that impacts first and the duration of the attack. J-8 defined several input rules which limit how missiles may be allocated. A mathematical model of the J-8 missile allocation problem was developed that uses a goal programming approach to solve the problem. The input rules defined the constraints for the problem. J-8 provided unclassified sample data to use as a test case. The model was used to solve the sample problem with nine different variations of the data. The model developed is a flexible tool designed to solve missile allocation problems whose objective is to minimize the first impact time or minimize the duration of the attack. The model uses binary variables, so it may be impractical to use if the number of binary variables gets large. However, in the foreseeable future, the size of the problem should decrease, thus making the model usable for at least several years.

14. SUBJECT TERMS
Allocations, Mathematical Programming, Goal Programming, Integer Programming

15. NUMBER OF PAGES
137

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL